uc3m | Universidad Carlos III de Madrid

Grado en Ingeniería Informática
2017/2018

Trabajo Fin de Grado

Análisis comparativo de diversos modelos de Deep Neural Networks aplicados al reconocimiento de imágenes

Marcos Minaya Montalvo

Tutor

Juan Manuel Alonso Weber

25 de Septiembre de 2018

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES

1. SUMMARY

1.1. Introduction

In 1940, the term artificial intelligence was coined at a conference in Dartmouth. Artificial intelligence is not something in itself. It can be considered as a set of algorithms in charge of emulating human reasoning when performing tasks. However, it is still far from that reality. While it is true that much progress has been made in this technological branch, there is much work to be done to imitate the way that the human brain functions.

Two key causes in the advance of artificial intelligence were the exponential growth of data and the advance of computational power. In the past, there was not enough data of enough power to perform task of this nature, but nowadays, the human being has tool and data to carry out numerous artificial intelligence assignment.

The huge amount of input data added to the large number of mathematical operations required made, in the 80's, that machine learning was an unworkable task. In those years the neural networks that were designed had only one layer, the computing capacity was infinitely less and we did not have the amount of data available to us today. Nowadays all these factors are no longer a problem.

On the other hand, it has been possible to verify that algorithms designed for the same tasks as neural networks have limitations. They do not have such a large generalization capacity and therefore they came up against a barrier when it comes to working, however, thanks to the RNAs, they have managed to cross that barrier and reach much further. The issue is not that deep learning techniques are very good at performing tasks of, for example, classification, but that they are becoming better than human beings themselves.

Within this set of learning algorithms, two large groups can be distinguished, supervised learning and unsupervised learning. The great difference between these two sets is not based on the fact that in the supervised human controls the entire learning process and in the unsupervised one does not. In the supervised learning, the expected output associated with the input data is known, and therefore, the difference between the computed and the expected output can be calculated. In this way, the weights of the network can be adjusted to make the two outputs look as much as possible and reduce the error.

On the other hand, in unsupervised learning the expected output is not know, and therefore its task is to group the input data according to criteria generated in the computation process of the network, and not according to a predefined class. The case that has relation with this work is the supervised learning, because neural networks are based on the adjustment of weights of their connections using functions of error minimization calculated thanks to the expected output of the network.

Warren McCulloch y Walter Pitts began conducting studies based on neural networks. There were two important aspects of the study. One based on the application of neural networks in the scientific field, and another which sought the biological understanding of the brain. This case is related to the first of these neural networks applications.

There are numerous applications related to the artificial neural networks, but the main objective of this work will be the classification of images. From the use of simple neuronal models to be able to classify and differentiate photographs of cats and dogs, to networks capable of identifying if, by means of an X-ray image taken from a patient, he suffers from some disease or not. A clear example of this is DeepMind, the artificial intelligence department of Google, which has been able to obtain impressive results when it comes to detecting eye diseases.

The programming of a neural network is based solely on the structure of its architecture and a series of functions and algorithms that make it have tools to learn. However, they are not programmed explicitly to learn what is imposed, but they lean by themselves, without the need of code lines that imposed it. The key of this is that, using the same architecture, one of these models is able to extract image patterns from different domains.

¿What the artificial neural networks are?

Basically, artificial neural network is a conceptualization of the natural behavior of the human brain, and how it processes information. Artificial neural connections, which are biologically known as synapses, are responsible of the network interconnection and their independent calculation modules (simple perceptrons), work together.

Like any machine learning method, deep learning is based on the presentation of different and numerous patterns to the model to train. When speaking of input patterns, reference is made to the data that is given as input to the model so that it is capable of extracting characteristics from them. As an example, you can understand as input patterns images that are given to the ANN, and the feature to expect from the network will be that it identifies that the image is a car.

Normally, ANN conception is formed mainly by 3 differentiated zones:

- **Input layer**: This layer is responsible of collecting data that comes from outside.
- **Hidden layer**/s: It is responsible for performing the relevant calculations in order to extract the patterns that relate the input data to the output of the network.
- **Output layer**: It is responsible for sending out the calculated output by the hidden layer.

¿What is the problem with ANN?

The big problem of ANN, is the black box behavior that characterizes these models. This behavior is a big problem for developers and engineers, because what happens is that these models work very well and give us very satisfactory results, but it is not known in detail how it has been able to reach them. We know the mathematical formulas that make possible the algebraic operations to be able to work with the input data, the disposition of the neurons in the network and all its parameters is known, but it is very complicated to monitor and analyze each one of the actions that lead to to arrive at the result that finally returns us.

For example, when it comes to distinguishing between pictures of cars and trucks, a deep neural network does not have any problem, however, it is not known if for this the network fixes its attention on the wheels, on the hood, on the windows or in any other characteristic of the vehicles.

This is the basic architecture of what is called multilayer perceptron. This network was designed as a universal approach. This name that was given to the model means that, theoretically, can approximate any non-linear function that relates the input data with the output. However, this model has limitations when it comes to exploiting the search space, being the local minimums in it its biggest enemies. This type of models is not usual, since neural architectures are created based on a specific type of problem, and not for any type of problem, as is the case of MLP.

But there are more types of neural architectures. One of the most outstanding, and which will also be analyzed throughout this study will be the convolutional neural network. Convolutional networks were not created as universal approach, they have, unlike the MLP (multilayer perceptron) a specific architecture for the datasets with which it is expected to work.

One of the qualities to stand out the CNN (convolutional neural networks) is the extraction of characteristics they perform. As previously mentioned, the neuronal application to be studied will be the classification of images, for which the convolutional networks are specifically structured. It must also be said that within the CNN architecture, which is more complex than that of a multilayer perceptron, the last part is formed by what is called a fully connected layer, which basically is an MLP.

There are a lot of datasets nowadays, but the study will focus on two mainly: MNIST and CIFAR10. Both are image datasets, but you can find many differences among them. The first one is the size of the input data. While an image of MNIST has a size of 28x28, which means a total of 784 pixels, in grayscale. This domain has 60000 images to perform the training and 10000 to test the model.

In the case of CIFAR10, the images have 3 layers of color, and their dimensions for each one are 32x32, which makes a total of 3072 entries. In this domain there are 50000 images for the training set and 10000 for the test set. As already mentioned, this type of problems are grouped into set of problems that can be addressed by supervised learning algorithms. Therefore, in both cases the inputs have an associated output which can be of 10 types.

Troughout this proyect, analyzes of the aforementioned models and their behavior with respect to the two domains described will be carried out. We will try to find out the biggest of the unknown that are planned in this work, why does the multilayer perceptron not obtain acceptable results against the CIFAR10 domain? Before carrying out the experiments, a study of the internal functioning of each model to be analyzed will be necessary.

1.2. Objectives

As has been well commented, the interest about the knowledge of neuronal models for image recognition and their classification is the challenge that is proposed throughout the following work. Two major objectives to be achieved though the analysis developed are:

- The study of the theoretical basis and internal functioning of the two proposed neuronal models, multilayer perceptron and convolutional network.

- Special emphasis on the operation of the MLP, in order to know the reasons why it is not able to achieve acceptable results with CIFAR10.

This last objective is so interesting, because there is not much information about it. With the study that will be execute, we expect to find answers or, failing that, certain ideas for the understanding of the internal process that an MLP executes, and why when working with two domains at the beginning so similar, it obtains distant results.

On the other hand we also want to investigate which models and architecture is the most appropriate for each domain. Although it has been shown that CNN obtain much better results when working with CIFAR10, their number of hidden neurons, the activation function types, error minimization algorithm, etc, are also parameters to analyzed.

Although much less important for this work, one of the objectives that were also proposed was the acquisition of knowledge with the tensorflow library. Because of all the progress in terms of artificial intelligence is concerned, tensorflow provides a very important value, especially in this types of task in which we want to deepen knowledge about neural networks.

Unlike other types of libraries such a Keras, tensorflow is aimed at a lower level of abstraction, being able to say that it is “a lower level library”. It is a great advantage when the purpose of the study is the deepest foundations of these networks, however, the cost of understanding the code, its flows of execution and data visualization have been more expensive.

1.3. Experimentation

In this section we will carry out a comparative analysis between the differences in terms of results and performance of the two types of networks with which we have worked in the previous sections. As it has been highlighted in previous occasions, the motivation of this work was to know the reasons why the MLP, even if it is a universal approach, does not manage to reach the success rates that CNNs reach. While it is true that the latter has a final layer which simulates the behavior of an MLP (fully connected layer), the results are frankly distanced.

Subsequently, a series of domains will be presented with which neural networks are trained for classification tasks. As far as machine learning is concerned, data is an essential part of the learning process, and therefore knowledge about the different dataset is fundamental. It is necessary to have notions of the type of data with which one is going to work, which must be numerical (as it is the case of the domains composed of images) because the networks of artificial neurons are only able to work with this type of values.

On the other hand, the organization of data in these domains is a crucial part. For example, the architecture of the network will be influenced by the dimensionality of the input data. The MNIST domain has a series of images which have a size of 28x28 pixels in gray scale, generating a total of 784 values for each image, and CIFAR10 these are 32x32 photographs in color, in RGB format (3 layers of color)), throwing a total of 3072 entries. In this way, the network that works with MNIST must have an input layer with 784 neurons, and if CIFAR10 is analyzed, it must include 3072 in that same layer. In turn, the output layer will also be influenced by the domain, because each of them has a total of 10 classes for classification, and therefore the networks must have 10 output neurons.

The way in which the analysis of the two models will be carried out will be through the exhaustive observation of the trend of different variables that each model returns. This is possible through checks on the evolution of the success rate in both the training process and the test. The training and test rates show the percentage of instances well classified in each of the two sets. Although both rates are important, the one that truly reflects if the model is functioning as expected is the test rate.

The training is based on presenting to the model instances with which you have already trained, and the test tests whether the model is able to correctly classify data with which the network has not trained, and therefore see if the model is capable to generalize.

While it is true that CNNs exceed 98% accuracy, their execution times are extremely high compared to those of the MLP. It would be necessary to put in a balance the time against the accuracy rate that you want to reach. When the results are observed using multilayer perceptron, this can reach 98% in test, but with a relatively low computational cost and time. Keep in mind that the amount of weights with which you work in an MLP is much lower than in a convolutional network.

To converge to its maximum precision, the cost of a CNN when facing MNIST is unbalanced. Is it really worth this type of networks for such a domain? Frankly, it does not seem that way, unless they are needed they reach somewhat higher hit rates, but they are not too differentiated. If we wanted to apply this domain to some real-world problem, such as the passage from handwritten text to plain text, of every 10,000 characters that the network receives, it will classify some 190 badly, against the approximately 230 that would confuse an MLP.

Entering the field of execution times does not make much sense when comparing MLP and CNN over CIFAR10. In the first place, the comparison between the success rates shown by each of the models is great. As it has been shown in the section of the analysis, the CNNs surpass the 84% of precision in test, rate which the MLP costs them to reach even in regard to training refers. The free code that tensorflow [27] shows us results of 86% of success rate in test. While it is true that convolutional networks add a much higher computational cost than MLPs, this is justified over domain, as with MNIST no.

Both models have significantly higher execution times compared to simpler domains, however, for the same number of cycles, the differences between success rates of one model and another are enormous. The time required by CNN to reach the highest rate achieved by the MLP is negligible, taking into account that, that the success rate does not exceed 60%.

1.4. Conclusions

The results obtained have been as expected in a certain way. As it was already known, the multilayer perceptron has a correct functioning when working with the MNIST dataset. However, it is not able to converge to an acceptable solution when talking about CIFAR10. Many of the efforts that have been invested in this experimentation have been directed towards the improvement of multilayer perceptron results.

On the other hand, although convolutionals have a lot of potential, it has been found that working with MNIST, despite having good results, consumes too many computational resources. It is also worth that the execution times of the CNN's are much higher than those of the MLP's in terms of MNIST. However, the use of CNN over CIFAR10 is more than justified, because although the execution times are very high, it obtains very satisfactory results.

As already mentioned before, numerous parameters have been analyzed to better understand the behavior of both models. One of the most important parameters is the learning rate. With an excessively high rate the model will be too random in its search and will not converge to any result, but a low rate will cause the network is not able to explore the search space therefore, not find the solution.

Although architecture is not as important as the learning rate, depends greatly on the success of the model throughput the experimentation. First hand, it should be noted that an initial architecture conditions experimentation. If this is not chosen properly, the task of experimentation can be somewhat difficult. If we focus on an MLP, it has been observed that with only one hidden layer, with a number of neurons between a range of 500 and 1000, it is sufficient to achieve very optimal results. In the case of the CNN, it was more optimal to use 3 hidden layers with 800 neurons each

Therefore, the research part according to the choice of the initial architecture paid off. One approach that could be successful was the "Thumb method", which advised the use of a number of hidden neurons approximately equal to two thirds of the total input volume.

If too many neurons are used, the model will take too long to converge, and with few neurons it will not even be able to do so. The more complex the model is, as it can be the case of the CNN, in which the amount of weights and parameters is very high, a much lower learning ratio is needed, to avoid large changes in those weights and favor the convergence to values of high success. However, in the case of the MLP, even if you work with one domain or another, you do not need such a big change between learning ratio, as when you compare an MLP with a CNN. While it is true that to work with the domain CIFAR10 it is necessary to reduce this ratio slightly, it is not as marked a difference as when passing a CNN analysis

An important mention must be made of a parameter that at no time was it intended to analyze: The standard deviation in the random initialization of weights within the MLP. There is nothing special about this parameter when talking about MNIST, but it is surprising to see how much weight it has in CIFAR10.

It was found that, while the MLP managed to learn with a rate of 0.001 over CIFAR10, by modifying the standard deviation increasing it, that same rate did not mean that the model was not able to advance.

Moreover, it was necessary to reduce it to less than 0.00001 for the model to be able to extract patterns, and even then it did not do it with the same efficiency as with smaller ranges of this standard deviation. This point could be the subject of a much deeper study, analyzing the weights and their evolution from the beginning with various types of standard deviation.

Regarding the size of the mini-batches, it has been observed that the use of large volumes of data as input does not contribute practically any value to the training process. Mini-batches hace a lower computational cost for each iteration of training, which makes training less expensive and requires less time. As far as issues of success in the set of test are concerned, the use of these mini-batches reaches hit rates equal or better than larger entries.

Finally add that, although it was an expected result, it seems that the MLP is the model indicated for the classification of images in MNIST, while the CNN work infinitely better with CIFAR10. It seems that it was a good idea to think, when designing a convolutional model, in the extraction of characteristics. It seems one of the biggest reasons why the MLP cannot extract just color image information.

If both domains are compared, it is obvious that the manuscript numbers have much more relationship between them than the images of cars, cats, airplanes, etc. It means that MNIST is a much simpler dataset, and therefore MLPs too.

Numerous efforts have been made to ensure that the multilayer perceptron is capable of reproducing the behavior of CNNs when working with a domain such as CIFAR10. Despite all the study and analysis carried out, no clear conclusion has been obtained to affirm that a CNN is capable of improving the way it does with the results of the MLP. The problem of an artificial neural network acquiring the behavior of a black box, you cannot know exactly what happends inside, makes the understanding of the causes why it works or stop working it is very expensive.

Despite not having obtained the same behavior on CIFAR10 with MLP as with CNN, they have been able to improve their results. In the first experiments, although not reflected in this work since they were simple initial tests, it was not possible to overcome a 47% success rate. Taking into account these data, a maximum success of more than 67%, and a convergence of an average of 62% make the efforts invested have paid off.

So it would be correct to say that an MLP is a universal approximator? It may be that, initially, when it was designed back in the 70's in which the amount of dataset and problems to be addressed was not so great, if it could conceive the idea that an MLP could find a solution to any non-linear problem in a theoretical way. However, when you want to apply this type of models on the real world, the previous statement comes down.

The MLP was a good base to begin the development of networks of artificial neurons, and numerous architectures, as well as CNN can incorporate this model. However, it has been found that computational power and problem solving was left behind, giving way to a new range of networks.

1.4.1. Future work based on the conclusions

Throughout the study, numerous parameters have been found that could be of interest for a future analysis. One of them is the aforementioned standard deviation. The variation in this parameter with the MLP on CIFAR10 had a very significant influence on the start of training. A much more in-depth study could be conducted to understand the impact of this parameter on artificial neuron networks.

"Neuronal death", has been a concept that has been tried to avoid by the function Leaki_ReLU. It would be interesting from data extracted from the models, such as the Sparsity, to carry out a deeper study avoiding to a greater extent this problem so common in the network of artificial neurons.

All this added to the use of GPUs, which greatly reduce the execution time can lead to a second study analyzing these and other parameters of great interest extracted throughout the investigation.

2. INTRODUCCIÓN

Hoy en día es importante entender el papel que juega a nivel tanto tecnológico como económico el terreno de la inteligencia artificial. El machine learning no es un concepto nuevo, sin embargo, está teniendo un auge destacable en los últimos años.

¿Pero que es el Machine Learning? El Machine Learning es una de las ramas de la inteligencia artificial la cual se encarga de generar software que sea capaz de generalizar a partir de una serie de datos recibidos. Otra definición que encaja muy bien con este concepto es la que dio Arthur Samuel en el año 1959 , el cual dijo que el Machine Learning es "un campo de estudio que da a los ordenadores la habilidad de aprender sin la necesidad de ser explícitamente programados"[1].

Si se simplifica mucho las tareas que son capaces de realizar este tipo de software podremos encontrar principalmente 3 tareas:

- Regresión: Método utilizado para la predicción del valor de un atributo continuo
- Clasificación: Método utilizado para la predicción del resultado de un atributo con un valor discreto dadas una serie de características.
- Agrupación: Método utilizado cuando se necesita clasificar una serie de instancias de datos entrantes de las cuales no se conocen previamente las categorías.

Dentro del ámbito del Machine Learning, a su vez, se pueden agrupar los algoritmos en dos grandes grupos, el aprendizaje no supervisado y el aprendizaje supervisado:

Aprendizaje supervisado: Se basa en el entrenamiento de un modelo, en el que los datos están clasificados o etiquetados, extrayendo patrones de ellos. Por lo tanto al presentar un nuevo dato, el modelo será capaz de identificar a que clase pertenece.

Aprendizaje no supervisado: En el aprendizaje no supervisado no se cuenta con datos previamente etiquetados, y por lo tanto trata de extraer características del conjunto de datos con el fin de poder organizarlos de alguna manera.

Como un nivel más bajo de abstracción dentro del propio machine learning encontramos la definición de deep learning, la cual va mas enfocada a lo que se va a desarrollar en el siguiente trabajo. El deep learning o aprendizaje profundo, utiliza diferentes estructuras de redes neuronales para, mediante sucesivas capas neuronales, lograr el aprendizaje de las mismas o

extracción de patrones. El nombre de deep learning viene precisamente por eso, por la utilización de sucesivos niveles de profundidad de capas de representaciones de datos.

Como cualquier método de machine learning, el deep learning se basa en la presentación de diferentes y numerosos patrones al modelo a entrenar. Cuando se habla de patrones de entrada se hace referencia a los datos que se dan como input al modelo para que este sea capaz de extraer características de los mismos. Como ejemplo, se puede entender como patrones de entrada imágenes que se le den a la RNA, y la característica a esperar de la red será que esta identifique que la imagen es un coche.

Toda esta serie de tareas que hacen posible que un ordenador "aprenda" vienen dadas gracias a la aplicación de las matemáticas siendo el Machine Learning el caso en el que estas matemáticas vienen reflejadas por operaciones matriciales (Algebra Lineal).

Pero el deep learning no es más que una agrupación de algoritmos destinados a tareas de inteligencia artificial, que vienen directamente relacionados con las redes neuronales. Estas redes neuronales, como se mostrará mas adelantes, son una conceptualización del modelo biológico de la corteza cerebral. Las neuronas vendrían representadas por lo que se denominan perceptrones simples, de modo que cuando dichos perceptrones se conectan e interactúan entre ellos adquieren el nombre de perceptrón multicapa, el cual tenía en un principio como afán simular el comportamiento del cerebro humano.

¿Pero porque este auge del deep learning? Pues bien, como ya se ha comentado antes, para que una red neuronal tenga unos resultados aceptables, es necesario utilizar una cantidad de datos muy abundantes, los cuales posteriormente serán tratados mediante operaciones algebraicas, sobre todo matriciales, para poder extraer patrones de ellos y así "aprender".

Esta cantidad ingente de datos de entrada sumado al gran número de operaciones matemáticas necesarias hacía que, en la época de los 80's, fuera una tarea poco viable. Por aquellos años las redes neuronales que se diseñaban apenas tenían una sola capa, la capacidad de cómputo era infinitamente menor y no se contaba con la cantidad de datos de los que disponemos hoy en día. Hoy en día todos esos factores ya no son un problema.

Por otro lado se ha podido comprobar que los algoritmos diseñados para las mismas tareas que las redes neuronales tienen limitaciones. No tienen una capacidad de generalización tan grande y por lo tanto se topaban con una barrera a la hora de trabajar, sin embargo gracias a las RNA se ha conseguido atravesar esa barrera y llegar mucho más allá. La cuestión no es que las técnicas de deep learning sean muy buenas realizando tareas de, por ejemplo clasificación, sino que están llegando a ser mejores que los propios seres humanos.

El gran problema de las RNA, el cual ha sido la motivación de este trabajo, es el comportamiento de caja negra que caracteriza a estos modelos. Este comportamiento es un gran problema para desarrolladores e ingenieros, ya que lo que sucede es que estos modelos funcionan muy bien y nos dan unos resultados muy satisfactorios, pero no se conoce en detalle como ha sido capaz de llegar a ellos. Se conocen las formulas matemáticas que hacen posibles las operaciones algebraicas para poder trabajar con los datos de entrada, se conoce la disposición de las neuronas en la red y todos sus parámetros, pero es muy complicado monitorizar y analizar cada una de las acciones que llevan a cabo para llegar al resultado que finalmente nos devuelve. Por ejemplo, a la hora de poder distinguir entre fotografías de coches y camiones, una red neuronal profunda no tiene ningún problema, sin embargo, no se sabe si para ello la red fija su atención en las ruedas, en el capó, en las ventanillas o en cualquier otra característica de los vehículos.

En cuanto a las redes neuronales, hay una gran variedad de modelos, dependiendo del tipo de problema que se quiera abordar, se utilizan unas u otras. Sin embargo, el primer modelo desarrollado de red neuronal fue el perceptron multicapa, anteriormente mencionado. Este es un conjunto de perceptrones simples interconectados entre ellos. Gracias a ese trabajo en conjunto de los perceptrones simples, el perceptron multicapa, es capaz de encontrar soluciones a problemas que no sean lineales.

Este modelo fue concebido como un aproximador universal. Es decir, debería ser capaz, con mayor o menor dificultad, de aproximar relaciones no lineales entre los datos de entrada y salida de cualquier problema. Esta afirmación tendrá gran importancia a lo largo del desarrollo de este estudio.

Por otro lado, este tipo de modelos no suelen ser usuales, ya que como se ha mencionado, las arquitecturas neuronales se crean en base a un tipo de problema específico, y no para cualquier tipo de problema, como es el caso del MLP.

Este es el caso de las redes convolucionales o CNN. Estas fueron ideadas con el fin de realizar, sobre todo, clasificación de imágenes, y para ese tipo de problemas funcionan realmente bien. También hay que decir que dentro de la arquitectura de las CNN, la cual es más compleja que la de un perceptrón multicapa, la ultima parte está formada por lo que se llama una capa totalmente conectada, la cual básicamente es un MLP.

3. DESCRIPCIÓN DEL TRABAJO A REALIZAR

En este trabajo se va a abordar un análisis del comportamiento del perceptrón multicapa frente a las redes convolucionales en dos dominios de imágenes distintos: MNIST y CIFAR10.

La cuestión es que, aunque haya redes que tengan una arquitectura específica para un conjunto de problemas en concreto, el perceptrón multicapa, al ser un aproximador universal, debería ser capaz de llegar a una solución aceptable para cualquiera de esos problemas. Sin embargo se ha comprobado que, aunque obtenga resultados bastante óptimos con el dominio MNIST, no es capaz de converger en una solución óptima para el CIFAR10, mientras que las CNN superan ampliamente la precisión que otorgan los MLP's.

Para poder realizar los experimentos necesarios y conocer de una manera más profunda el funcionamiento de los modelos neuronales a analizar, se realizará un estudio previo del funcionamiento de cada uno de ellos, incluyendo la base matemática que se encarga del aprendizaje de los mismos.

Posteriormente se presentarán una serie de dominios con los cuales son entrenadas las redes neuronales para tareas de clasificación. En cuanto al machine learning se refiere, los datos son parte esencial en el proceso de aprendizaje, y por lo tanto el conocimiento sobre los diferentes dataset es algo fundamental. Es necesario tener nociones del tipo de datos con el que se va a trabajar, los cuales han de ser numéricos (como es el caso de los dominios compuestos por imágenes) debido a que las redes de neuronas artificiales solo son capaces de trabajar con este tipo de valores. En caso de no ser así, como pueda ser en el tratamiento de textos, sería necesaria la conversión del texto a datos numéricos para que estos sean entradas validas de la red. Por otro lado, la organización de los datos en dichos dominios es parte crucial. Por ejemplo, la arquitectura de la red se verá influida por la dimensionalidad de los datos de entrada. El dominio MNIST cuenta con una serie de imágenes las cuales tienen un tamaño de 28x28 pixeles en escala de grises, generando un total de 784 valores por cada imagen, y e CIFAR10 estas son fotografías de 32x32 a color, en formato RGB (3 capas de color), arrojando un total de 3072 entradas. De este modo, la red que trabaje con MNIST deberá tener una capa de entrada con 784 neuronas, y si se analiza CIFAR10 deberá incluir 3072 en esa misma capa. A su vez la capa de salida también se verá influida por el dominio, debido a que cada uno de ellos cuenta con un total de 10 clases para su clasificación, y por ende las redes deberán contar con 10 neuronas de salida.

Las siguientes decisiones en cuanto a la arquitectura de los dos modelos se explicarán de manera más exhaustiva más adelante en el apartado 13. Del lado del MLP se ha optado por la estructura clásica. Una red neuronal completamente conectada con una capa de entrada, una capa oculta, y

una capa de salida. En lo que a la CNN se refiere, se utilizará una arquitectura con dos capas convolucionales, y una capa totalmente conectada. El resto de parámetros y variables que se utilizarán se explicarán más adelante –Apartado 13-.

El modo en la que se van a realizar los análisis de los dos modelos será mediante la exhaustiva observación de la tendencia de diferentes variables que cada modelo devuelve. Esto es posible mediante comprobaciones en la evolución de la tasa de acierto tanto en el proceso de entrenamiento como en el de test. ¿Que representan estas dos tasas? Las tasas de entrenamiento y de test muestran el porcentaje de instancias bien clasificadas en cada uno de los dos conjuntos. Aunque ambas tasas son importantes, la que verdaderamente refleja si el modelo está funcionando como se espera es la tasa de test. La de entrenamiento se basa en presentarle a modelo instancias con las que ya ha entrenado, y la de test comprueba si el modelo es capaz de clasificar correctamente datos con los que la red no ha entrenado, y por lo tanto ver si el modelo es capaz de generalizar. Una alta tasa de acierto en test quiere decir que el modelo es capaz de clasificar bien datos que nunca antes ha computado, y eso es lo que se busca con estos modelos.

Otra variable que se va a analizar en ciertos experimentos será el cross entropy, la cual vendrá explicada en profundidad –apartado 13-. El análisis de sus gráficas nos puede dar una idea de cómo se comporta un modelo.

Tras una serie de experimentaciones, y en base al objetivo que tiene este estudio, se realizarán las comparaciones y conclusiones oportunas para poder definir, dentro de cada ámbito, que arquitectura es la más recomendada.

4. MOTIVACIÓN

A lo largo de los últimos años, el tamaño de las bases de datos está creciendo de manera exponencial y a un ritmo vertiginoso. Uno de los conceptos que cada vez está más arraigado en la sociedad y coge fuerza día a día es el BigData [2]. Este término refleja la gran acumulación de datos que se generan minuto a minuto por cada uno de los usuarios en internet. Uno de los datos recogidos más impresionantes acerca del BigData es que, en los dos últimos años el ser humano ha generado más datos que en todo el resto de la historia de la humanidad.

En la siguiente grafica se puede apreciar el crecimiento de los datos a nivel mundial. Para que el lector se haga una idea aproximada, un exabyte de datos equivale a 10e18 bytes, es decir, 1.000.000.000.000.000.000 bytes de datos. Como se puede comprobar, se estima que para el año de o 2020 habrá más de 50.000 exabytes de información no estructurada en el mundo (Datos extraidos de la referencia [3]). Hay que destacar que el volumen de estos datos no estructurados crece mucho más rápido que los datos estructurados, lo que lleva a pensar que será necesario la generación de algoritmos o procesos de machine learning capaces de lidiar con esta información, la cual hay que decir, es notablemente más compleja de tratar.

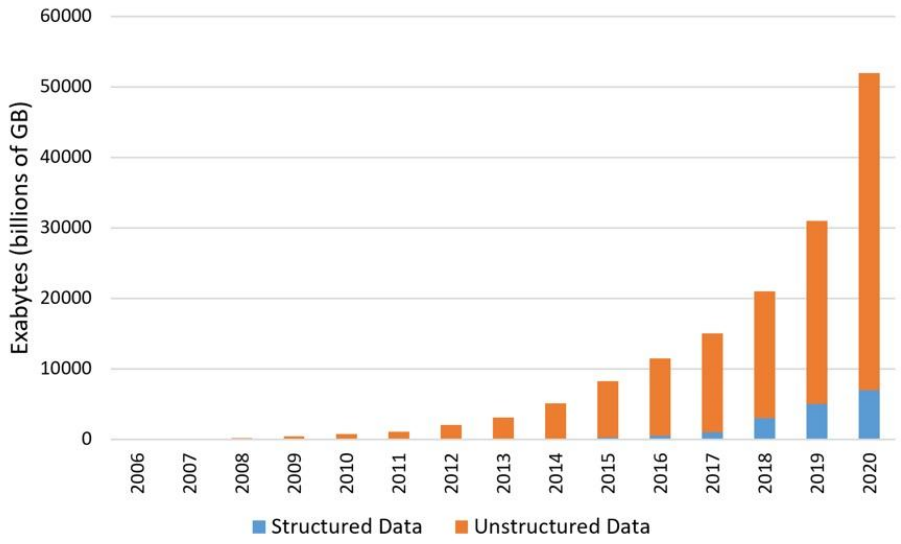

Figura 1 . Crecimientos de datos [3]

Por otro lado, aunque el crecimiento de datos sea exponencial, la potencia de computación con la que cuenta el ser humano también crece año a año. Aunque esta no se incremente de la misma manera, su aumento también es destacable. Gordon Moore publicó, en base a observaciones realizadas en la década de los 70's, la ley de Moore [4]. Esta afirmaba que la cantidad de transistores por chip de silicio de duplica cada año. Si bien estos chips no albergaban exactamente el doble de transistores cada año, fue una aproximación la cual hoy en día sigue teniendo mucho peso.

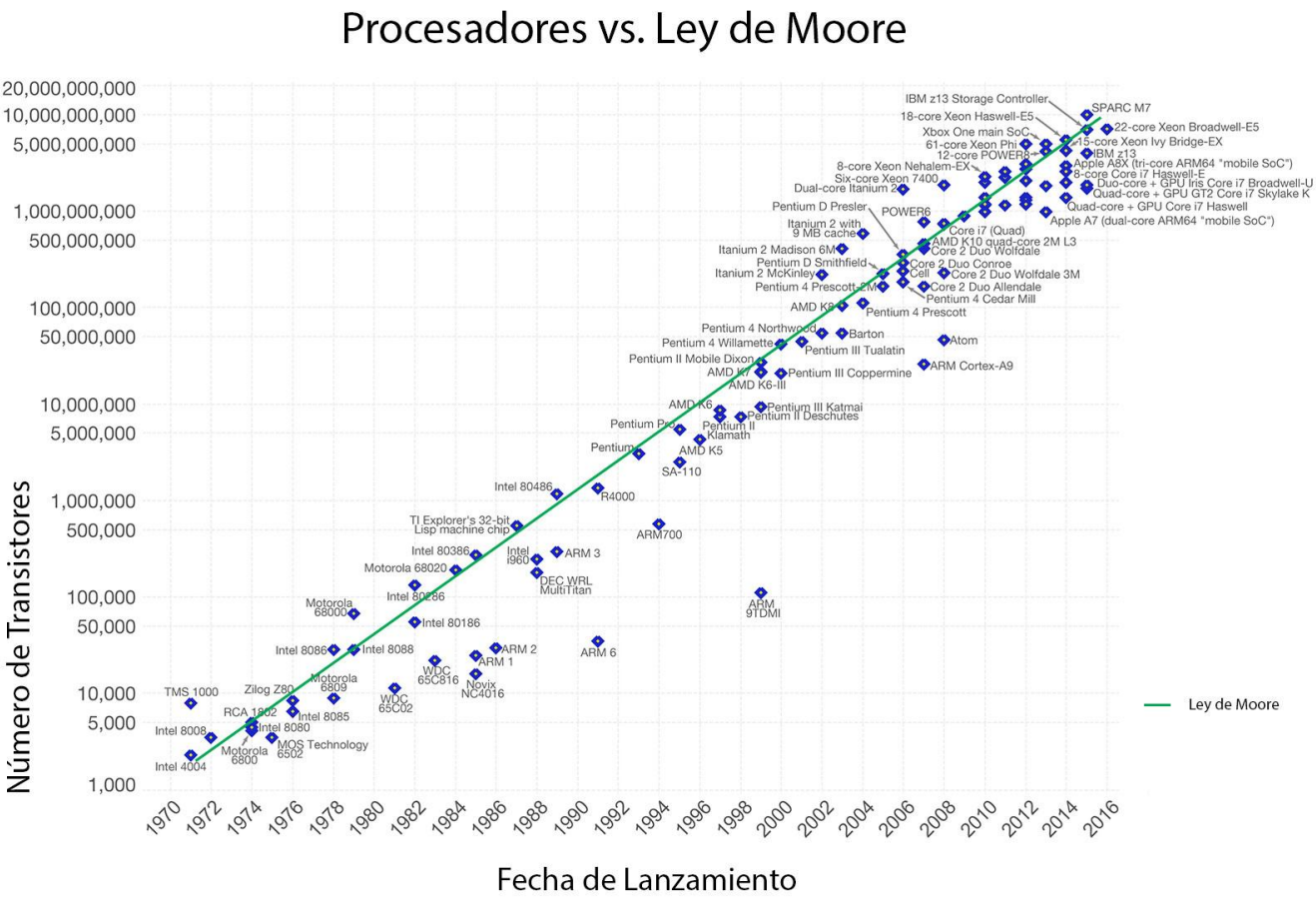

Figura 2. Ley de Moore [5]

Esta combinación de datos y potencia computacional hace que la inteligencia artificial, la cual en la década de los 80 era un sueño aun por alcanzar, hoy en día se pueda convertir en una de las herramientas más poderosas que existen. De todos modos, conociendo el potencial que puede llegar a alcanzar la inteligencia artificial, ha día de hoy se podría decir que esta en un estado embrionario. Sin embargo, dejando a un lado la juventud del machine learning, se puede

observar como el mismo sirve a numerosas empresas como un soporte vital, e incluso acompaña a la gente de a pie día a día en su vida cotidiana.

En lo que al ámbito empresarial se refiere, la inteligencia artificial brinda numerosas ayudas a multinacionales, otorgando así a estas empresas una ventaja competitiva respecto a las demás. Se pueden observar como estas nuevas tecnologías por ejemplo, utilizando bases de datos de años de idealización y trabajo con clientes, consiguen extraer información la cual usan para poder ofertar nuevos productos a clientes, obteniendo de este modo resultados mas que satisfactorios. Uno de los ejemplos que se pueden ver hoy en día es el CLV (Customer Livetime Value [6]) o valor de vida de cliente. Es un indicador muy utilizado por el machine learning en el contexto empresarial, y básicamente es un indicador de la cantidad de dinero que un cliente puede llegar a invertir en un periodo determinado. Este tipo de información, la cual se puede extraer de las bases de datos con las que cuenta una empresa, combinada con técnicas de inteligencia artificial, es un arma muy potente, que puede otorgar datos de como poder fidelizar a clientes y mejorar su experiencia con la marca. Este tipo de algoritmos suelen estar basados en un aprendizaje no supervisado, en el que no se tiene clara la clasificación de los clientes a los que se les quiere ofertar ciertos servicios.

Fuera del ámbito empresarial, y como tema a tratar en el estudio que se va a realizar, una de las aplicaciones con más fama por parte de la inteligencia artificial es la clasificación de imágenes mediante algoritmos basados en redes neuronales. Gracias a todo ese volumen de datos con los que se cuenta hoy en día, la clasificación de imágenes mediante computadores ya no sigue siendo un sueño, sino que ha pasado a ser una realidad.

Desde el uso de sencillos modelos neuronales para poder clasificar y diferenciar fotografías de peras y manzanas, hasta redes capaces de poder identificar si, mediante una imagen de rayos X tomada a un paciente, este padece alguna enfermedad o no. Un claro ejemplo de ello es DeepMind, el departamento de inteligencia artificial de Google, el cual ha sido capaz de obtener impresionantes resultado a la hora de la detección de enfermedades oculares [7].

La enorme potencia de estos modelos hace que el creciente interés por su estudio y comprensión sea del todo entendible, aunque esta tarea no sea sencilla. Cada vez son más los científicos de datos, matemáticos, estadísticos, físicos etc, que se introducen en el mundo de las redes neuronales para ser capaces de entender su funcionamiento interno, y esto es, en mayor medida, lo que impulsó al desarrollo de este trabajo

5. OBJETIVOS

Como bien se ha comentado, el interés acerca del conocimiento de modelos neuronales para el reconocimiento de imágenes y su clasificación es el reto que se propone a lo largo del siguiente trabajo. Cabe destacar dos grandes objetivos a alcanzar mediante el análisis desarrollado:

- Estudio de la base teórica y del funcionamiento interno de los dos modelos neuronales propuestos, Perceptrón multicapa y redes convolucionales.
- Especial hincapié en el funcionamiento del MLP, para de este modo conocer los motivos por los que no es capaz de alcanzar resultados aceptables con CIFAR10.

Este último objetivo es de especial interés, debido a que no se encuentra demasiada información al respecto. Con el estudio que se pasará a realizar se esperan encontrar respuestas o en su defecto, ciertas ideas para el entendimiento del proceso interno que ejecuta un MLP, y el porqué al trabajar con dos dominios en un principio tan similares, obtiene resultados tan distantes.

Por otro lado también se quiere indagar en qué modelo y arquitectura es la más adecuada para cada dominio. Aunque se ha demostrado que las CNN obtienen mucho mejores resultados al trabajar con CIFAR10, su número de neuronas ocultas, los tipos de funciones de activación, algoritmos de minimización de error etc. también son parámetros a analizar.

Aunque con mucha menos importancia para este trabajo, uno de los objetivos que también se proponía era la adquisición de conocimientos con la librería tensorflow. A causa de todo el avance en cuanto a la inteligencia artificial se refiere, tensorflow aporta un valor muy importante, sobre todo en este tipo de tareas en las que se quiere profundizar acerca del conocimiento de las redes neuronales. A diferencia de otro tipo de librerías como Keras, Tensorflow se dirige a un menor nivel de abstracción, pudiendo decir que es “una librería de más bajo nivel”. Es una gran ventaja cuando el propósito del estudio es las bases más profundas de estas redes, sin embargo, el coste de comprender el código, sus flujos de ejecución y visualización de datos han sido más costosos.

6. ESTRUCTURA DEL DOCUMENTO

Se ha decidido distribuir y organizar el documento de la siguiente manera:

- **Apartado 2**, se puede encontrar una breve introducción al mundo de la inteligencia artificial y redes neuronales.
- **Apartado 3** se explicarán los estudios previos y la consiguiente experimentación y su posterior análisis de los modelos neuronales con los que se va a trabajar.
- **Apartado 4**, mostrará las motivaciones que han hecho que se lleve a cabo este trabajo de investigación.
- **Apartado 5**, se introducirán de manera muy breve los distintos objetivos que se desea alcanzar con este estudio.
- **Apartado 7**, se realizara un análisis histórico de las redes neuronales, como fueron estas concebidas y sus aplicaciones.
- **Apartado 8,** se explican las dos grandes clasificaciones existentes en lo que aprendizaje se refieren.
- **Apartado 9,** se realizará un análisis en profundidad de los modelos neuronales con los que se va a trabajar.
- **Apartado 10**, se expondrán los dos datasets con los que van a ser entrenados los modelos anteriormente mencionados.
- **Apartado 11,** se explican las razones de la elección del lenguaje de programación y sus *framework*
- **Apartado 12,** A la hora de realizar experimentaciones se tiene que partir de una arquitectura y unos parámetros iniciales. En este apartado se expondrán las razones de la elección de dicha arquitectura y parámetros.
- **Apartado 13**, se mostrará todo el proceso que se ha llevado a cabo a lo largo de la experimentación, tanto la carga de datasets y la experimentación con ambos modelos.
- **Apartado 14**, en este apartado se realizará la comparación entre los resultados obtenidos por ambos modelos.
- **Apartado 15**, aquí se extraerán las conclusiones asociadas a la comparativa anteriormente mencionada.
- **Apartado 16 y 17**, se pasará a mostrar la planificación estimada que tuvo el proyecto en un inicio y la planificación que finalmente se ha llevado a cabo así como el presupuesto del trabajo realizado.

7. ESTADO DEL ARTE

El ser humano, a lo largo de la historia, siempre ha tenido el afán de mejorar sus condiciones de vida, dejando en un segundo plano la fuerza física para pasar a procesos con un nivel de tecnificación más alto, los cuales hace años requerían de mucho tiempo para que fueran realizados de manera óptima. Así pues, la creación de computadoras y otras tecnologías han contribuido de manera muy importante a la hora de la resolución y optimización de problemas que anteriormente no tenían respuesta o simplemente no se habían resuelto de manera eficiente.

Uno de los grandes retos en computación ha sido el diseño y construcción de máquinas capaces de realizar tareas que realizasen humanos, pero de manera más optimizada. De estos inventos se han podido definir las líneas para la obtención de maquinaria dotada de cierta inteligencia, en un intento por simular el comportamiento humano en ciertas tareas.

Estas fueron las bases que dieron lugar a lo que hoy se conoce como Inteligencia Artificial, y por consiguiente a una de sus ramas más importantes como son las redes de neuronas artificiales.

7.1. Base Biológica

Es importante que se tenga en cuenta, que como en numerosas tecnologías y ámbitos, las redes de neuronas artificiales tienen una base bilógica que impulsó este nuevo campo. La idea inicial de que se podía imitar el comportamiento del cerebro humano mediante un modelo matemático parecía algo un tanto fantasioso. Sin embargo este concepto no pudo estar más acertado y dio lugar a una herramienta con numerosas aplicaciones, y que aun hoy en día tienen mucho potencial por ser descubierto.

Se puede observar en la Figura 1 la estructura básica de una neurona biológica. En dichas neuronas la información recorre las dentritas hasta el axón, atravesando el soma. Todo flujo de datos que recorre las neuronas del cerebro humano tiene que pasar a través de ellas en forma de impulsos nerviosos. Hay estimaciones del numero de neuronas alojadas en nuestro cerebro que sugieren que hay un total de 10*10^10.

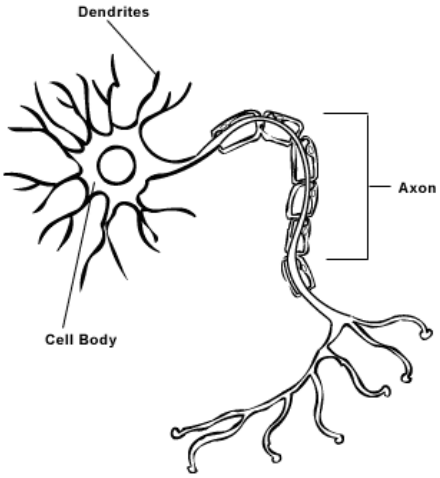

Figura 3. Neurona biológica [8]

En cuanto al funcionamiento de la red, el procesamiento de la información sigue un circuito más o menos estándar. Se transmiten impulsos nerviosos, que viajan por toda la neurona comenzando por las dentritas hasta llegar a los botones terminales. Las conexiones que existen entre las neuronas se denominan sinapsis, las cuales son direccionales. A nivel cerebral se puede observar que una red de neuronas tiene una organización horizontal en forma de capas y una en forma de columnas en el eje vertical –Figura 4-.

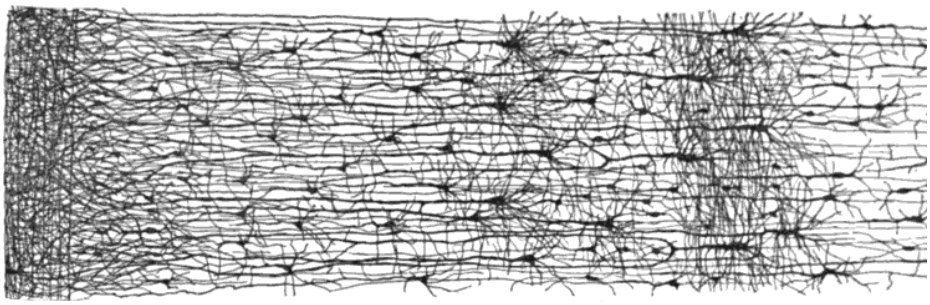

Figura 4. Estructura neuronal del cerebro humano [9]

Toda esta base biológica sirvió de inspiración a numerosos científicos a lo largo de la historia:

- Walter Pitts y Warren McCulloch intentaron en el año 1943 realizar una conceptualización del cerebro humano mediante una red de neuronas conectadas entre si que eran capaces de la realización de operaciones de tipo lógico.

- 6 años más tarde, Donald Hebb explicó en su libro (The Organization of Behavior) el funcionamiento de las sinapsis de neuronas, lo cual relacionó con la ahora famosa regla de aprendizaje.

- En el año 1951, Minsky y Edmons intentaron simular el cerebro de una rata mediante más de 300 tubos de vacío y el sistema de pilotaje automático de un bombardero b-24. Esta red estaba compuesta por un total de 40 neuronas artificiales.

- En 1957, Frank Rosenblatt creó el perceptrón simple, unidad básica de cómputo de las actuales redes neuronales.

- James Anderson en los años 70 desarrolló lo que él llamaba como asociador lineal, el cual consistía en la suma de las entradas de diferentes integradores lineales.

- En el año 1982, un trabajo de John Hopfield describe, con una alta precisión matemática, un modelo de red neuronal.

- Unos años más tarde, en el 1988, se unieron la IEEE y la INNS (International Neural Network Society). Esta unión produjo la "International Joint Conference on Neural Networks". EN ella se llegaron a realizar hasta 430 artículos relacionados con redes neuronales.

7.2. Conceptualización del modelo biológico

El nombre que recibe la conceptualización del modelo biológico neuronal es RAN (Red Neuronal Artificial). Dicha conceptualización que se realizó se basa principalmente en unidades de procesamiento interconectadas de una manera densa, a las cuales llamaremos neuronas. Dichas neuronas, al igual que las alojadas en nuestro cerebro, reciben procesan y transmiten señales.

Se puede apreciar en la Figura 5 la comparativa entre las representaciones de interacción entre las neuronas bilógicas del cerebro humano, y su correspondiente red neuronal artificial en la que se simulan las entradas a la red, sus interconexiones y sus salidas.

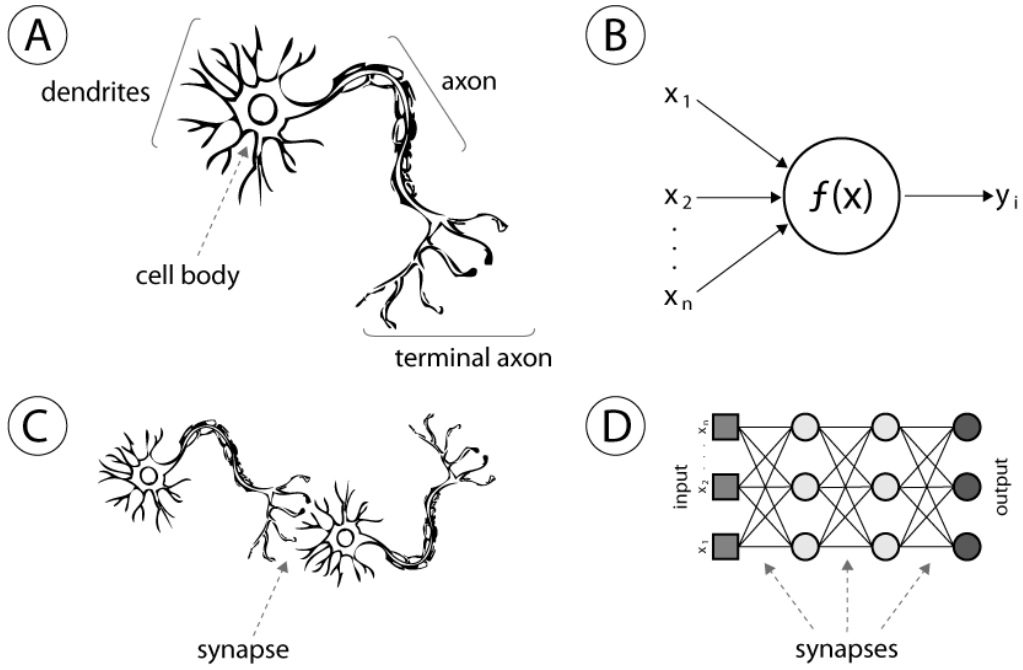

Figura 5. Conceptualización neuronal [10]

Las neuronas artificiales son pequeñas unidades de procesamiento, las cuales operan en paralelo y tratar de devolver una función mediante el ajuste de los pesos de la red. Estos pesos (Wi) están relacionados con las conexiones entre neuronas. Estas condiciones actúan bajo una cierta polarización bi conocida como "bias", sobre la función de activación i.

En las neuronas biológicas, la información de cada neurona es transmitida por medio de las dentritas, la cual será evaluada y procesada por el núcleo, y pasará por el axón para servir de entrada a otra neurona.

En una neurona artificial las entradas son denominadas Xi, y vienen representadas por vectores o señales. Los pesos Wi representarán la intensidad de la sinapsis entre neuronas. Tanto Xi como Wi son valores escalares.

7.3. Aplicaciones de Redes Neuronales

Las redes de neuronas artificiales tienen un tremendo potencial y versatilidad, y por lo tanto cuentan con un amplio abanico de aplicaciones. Hay que resaltar que las Redes neuronales ya han sido incorporadas a numerosos dispositivos, los cuales muestran muy buenos resultados a la hora del reconocimiento de patrones o distribución de datos.

Algunas de las aplicaciones que tienen estas redes hoy en día son:

- Reconocimiento de caracteres: Esta aplicación estuvo en auge hace algunos años. Hoy en día una de las tareas que se cubren en este ámbito es el reconocimiento de texto manuscrito.

- Compresión de imágenes [11]: Debido a que las redes neuronales son capaces de tratar grandes volúmenes de datos a la vez, hace que sean válidas para la compresión de imágenes, las cuales cada vez son más grandes y pesadas debido al aumento de la calidad.

- Valores en bolsa: Esta aplicación fue una de las primeras ideas a las que aplicar redes neuronales para el desarrollo de este estudio. La bolsa es algo muy complicado, en la que participan numerosas variables que hacen que una pequeña modificación en un valor pueda llegar a causar cambios más notables. Las redes neuronales son capaces de examinar un gran número de datos para después poder realizar un estudio de como fluctuará el mercado.

- Medicina: Esta es un área en la que las RNA han ganado mucha importancia. Por ejemplo, a la hora de la detección de afecciones cardiopulmonares. Estos modelos no sustituyen el trabajo de los médicos, sino que los ayudan a realizar diagnósticos.

De entre todas estas aplicaciones la que resultó de mayor interés fue la tiene relación con el reconocimiento de caracteres, y por ende, de la clasificación de imágenes. En apartados posteriores se analizarán dominios relacionados precisamente con esta aplicación de las redes de neuronas artificiales.

7.4. Entorno socio-económico

Al ser un trabajo centrado amplia y únicamente en la experimentación y el análisis hace que sea muy difícil identificar con exactitud un impacto socio-económico como tal.

El ámbito en el que se ha centrado todo el estudio ha sido en el reconocimiento de imágenes, una de las aplicaciones más utilizadas y en auge en los últimos años por parte de la inteligencia artificial, más en concreto las redes neuronales.

Por lo tanto el análisis y estudio de este tipo de aplicaciones podría tener algún tipo de impacto socioeconómico, teniendo algún valor para la comunidad científica a modo de poder realizar experimentaciones desde algún otro punto de vista o mejorar los resultados anteriormente obtenidos.

Cabe destacar que al final de este trabajo se ha incluido el presupuesto calculado en base a los costes de esta investigación.

7.5. Marco regulador

En cuanto al marco regulador se refiere, no hay mucho que decir. La totalidad de los datos, lenguajes de programación, plataformas online y frameworks son de licencia libre, y lo por lo tanto no están sujetos a ningún tipo de licencias privativas.

Cabe destacar las posibles aplicaciones que pudieran tener los resultados de esta investigación tuvieran un uso fraudulento. Sin embargo las motivaciones que han impulsado este estudio son meramente didácticas, y los dominios con los que se ha trabajado, siendo números manuscritos y fotografías a color (las cuales no están sujetas a ningún tipo de condiciones privativas) no pueden dar lugar a aplicaciones de uso nocivo.

8. APRENDIZAJE SUPERVISADO Y NO SUPERVISADO

Si se habla de machine learning, cabe destacar que este se divide en dos grandes grupos o tipos de problemas a abordar: aprendizaje supervisado y aprendizaje no supervisado. Cuando se habla de supervisado y no supervisado no quiere decir que el aprendizaje este bajo la vigilancia o no de un humano.

El aprendizaje en ambos casos se basa en el mismo principio: la extracción o cálculo de una función a partir de unos datos de entrada. La gran diferencia entre aprendizaje supervisado y no supervisado es el conocimiento que se tiene de los datos de entrenamiento a *priori*.

En el aprendizaje supervisado, los patrones de entrada se estructuran en pares, en los que uno de los componentes son los datos en sí, y el otro es el resultado deseado en base a esos datos de entrada. Esto se denomina tener los datos "etiquetados". Por lo tanto, lo que se busca con este tipo de aprendizaje, es la generación de la función, la cual sea capaz de predecir el valor asociado (etiqueta) en base a unos datos de entrada de los que se desconoce dicho resultado. Esto se consigue gracias a que el modelo en un inicio fue entrenado con datos de los cuales se conocía su salida, y el modelo extrae patrones para ser capaz de predecir dicha salida. De este modo, al introducir un nuevo dato, el cual nunca se usó en el entrenamiento del modelo, este abstrae esos mismos patrones y consigue calcular cuál sería su clase correspondiente. Normalmente este tipo de aprendizaje está muy relacionado con problemas de regresión y clasificación, siendo este último el que se abordará más adelante.

Dentro del aprendizaje supervisado se encuentran diferentes tipos de aprendizaje:

Aprendizaje por refuerzo: Este aprendizaje es uno de los más usados. Está basado en el uso de ejemplo de los que se cuenta con el comportamiento deseado, para que a la hora del entrenamiento, dichos comportamientos se refuercen de manera positiva, mientras que a los no deseados se les asigne un refuerzo negativo.

Aprendizaje estocástico: En este caso, al ser un aprendizaje supervisado, también se cuenta con una salida deseada, sin embargo, se realizan cambios aleatorios en los valores que se ajusten a la función que relaciona los datos de entrada y salida. De este modo se va observando el comportamiento del modelo en base al objetivo deseado y una serie de distribuciones de probabilidad.

Aprendizaje basado en la corrección del error: Este último es el que será utilizado en análisis de los diferentes modelos de redes neuronales. El aprendizaje por corrección de error se basa en el ajuste de los pesos de la red en función de la diferencia entre los valores deseado y los obtenido

como salida por la propia red. Como ya se mencionará más adelante, el aprendizaje de estos modelos consiste en una regla bastante sencilla, donde para cada neurona de la capa de salida se calcula la desviación a la salida objetivo, representando esta diferencia como un error.

En el caso del aprendizaje no supervisado, como ya se ha comentado anteriormente, no se cuenta con un conocimiento a *priori*. En ocasiones este tipo de aprendizaje también es denominado como aprendizaje auto-supervisado, debido a que la red no recibe ningún tipo de información por parte de su entorno que le pueda dar nociones acerca de si la tarea se está realizando de manera correcta. Se pueden decir que son redes que se auto organizan. De este modo, el modelo que trata unos datos en un aprendizaje no supervisado trata a estos mismos como un conjunto de variables aleatorias. Lo que se busca es la construcción de un modelo de densidad para dicho conjunto de datos. Una de las tareas asociadas a este tipo de aprendizaje es la de agrupamiento (clustering), en el que se busca generar agrupamiento basado en similitudes entre los datos.

Al igual que con el aprendizaje supervisado, en el aprendizaje no supervisado también se pueden encontrar diferentes tipos, y de entre ellos el más destacable es el aprendizaje Hebbiano [12]. En este modelo se pretende, calcular la familiaridad de los datos de entrada, hallando las correlaciones de los valores de las activaciones de las neuronas.

8.1. ¿Qué es una neurona artificial?

Una neurona artificial, comúnmente denominada perceptrón o perceptrón simple, es la unidad básica en una red neuronal, la cual basa su funcionamiento en una discriminación lineal. En base a esto se puede desarrollar un algoritmo que sea capaz de seleccionar un subconjunto a partir de un conjunto de elementos mayor. La gran limitación del perceptrón simple es que solo puede separar el espacio de búsqueda mediante un hiperplano, es decir, solo puede hacer una clasificación binaria de los datos de entrada. Por lo tanto, este modelo, podría simular puertas lógicas como AND y OR, pero no sería capaz de simular un módulo XOR, el cual partiría el espacio de búsqueda en 3 zonas.

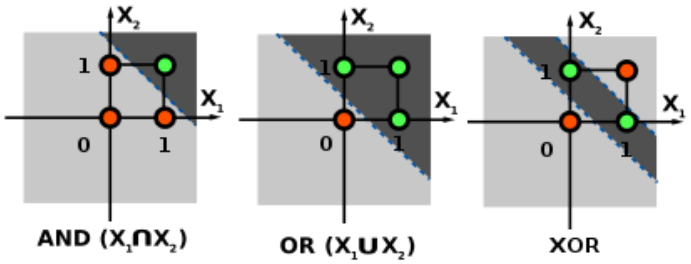

Figura 6. Funciones AND, OR, XOR

El funcionamiento interno de un perceptrón simple es muy básico. En su arquitectura cuenta con una serie de entradas Xi, unos pesos asociados a cada entrada Wi, un umbral U y la función f(x) la cual arroja un valor que sirve para clasificar a la entrada Xi como un caso positivo o negativo. De este modo un perceptrón simple tendría la siguiente arquitectura conceptualizada asociada a la función de activación f(x):

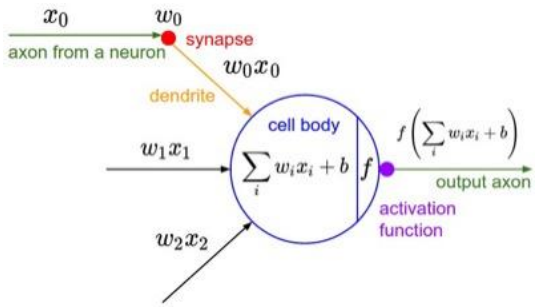

Figura 7. Perceptrón simple [13]

$$f(x)\begin{cases} 1 si x.w - u > 0 \\ 0 \text{ en e resto de casos} \end{cases}$$

La limitación anteriormente mencionada viene dada a la hora de calcular la salida de la neurona y su correspondiente error. Este tipo de modelos se basan en un aprendizaje supervisado y por lo tanto se sabe la salida esperada de la neurona para así poder hallar el error en base a la salida calculada. En el caso del perceptrón simple, el cálculo del error no se basa en decir en cuanto se ha equivocado el modelo, sino simplemente si se ha equivocado o no.

Unos años más tarde, en el 1960 Widrow y Hoff desarrollaron la idea del ADALINE. Este tipo de neurona tenía una estructura idéntica a la del perceptrón simple, sin embargo en este caso se utilizaba para el cálculo del error directamente la salida de la red, la cual es un valor real. De este modo se calculaba el error producido en la predicción. Este modelo servía a su vez para resolver problemas de regresión lineal.

8.2. ¿Qué es una red neuronal artificial?

Aunque hay un gran número de tipos de redes neuronales, una red neuronal convencional es un modelo de computación que se basa en la combinación de un gran número de perceptrones

simples, de forma que al trabajar en conjunto emulen el comportamiento de una red neuronal biológica.

Todas las neuronas dentro de una red se organizan en lo que se denominan capas, de las cuales se pueden distinguir 3 tipos principalmente:

- Capa de entrada: Se reciben los datos que proceden del entorno.
- Capa oculta: Se realizan los cálculos pertinentes asociados a la red neuronal.
- Capa de salida: Proporciona al exterior los resultados obtenidos de las capas ocultas.

Como se ha explicado anteriormente, el perceptrón simple era capaz de realizar clasificaciones binarias, y ADALINE podía modelar regresiones lineales. Sin embargo, con la combinación de neuronas simples con unas funciones de activación no lineales y leyes de aprendizaje, da como fruto un modelo que es capaz de aproximar relaciones no lineales entre variables de entrada y de salida.

Las redes neuronales son capaces de aprender ya que examinan los registros individuales, generando una predicción para cada uno de ellos y realizando los ajustes correspondientes, en base a esa predicción, de los pesos de la res cuando se realiza una predicción incorrecta. Es decir, al igual que en el modelo de ADALINE, se calcula un error con valor real, y se ajustan los pesos en base a ese error. El proceso de cálculo de error y ajuste de pesos se repite numerosas veces, de modo que la red vaya mejorando sus predicciones, hasta que se sea capaz de llegar a un error mínimo o muy pequeño.

9. MODELOS NEURONALES A ANALIZAR

9.1. Perceptrón multicapa (MLP)

El perceptrón multicapa (MLP) es uno de los más famosos tipos de red neuronal, basada en conexiones hacia adelante. Este modelo es una generalización de lo que se conoce como Perceptrón Simple, el cual no es capaz de solventar problemas con separación no lineal. Como ya se ha mencionado anteriormente, Minsky y Papert propusieron un modelo en el cual, mediante la combinación de estos perceptrones simples, se podían obtener soluciones para ciertos problemas no lineales. Posteriormente Rummelhart, Hinton y Wilians, usaron la idea de Minsky y Papert para idear una manera de retro propagar errores hacia neuronas ocultas, dando lugar a la regla denominada regla delta generalización y, por ende, al MLP que conocemos en nuestros días.

El MLP es una red neuronal de propósito universal, es decir, cualquier función continua, en un espacio R^n puede ser aproximada mediante un perceptrón multicapa. Por otro lado, esto no implica que el MLP sea una de las redes más potentes, debido a que cuenta con una serie de limitaciones. Por ejemplo, el proceso de aprendizaje busca en un amplio espacio de funciones, una que aproxime las entradas y salidas del problema, lo cual complica el aprendizaje y reduce drásticamente su efectividad, y es este precisamente el problema que se tratará más adelante cuando se trabaje con dominios relativamente grandes.

9.1.1. Arquitectura

La arquitectura del MLP viene dada por una disposición de neuronas en capas a diferentes niveles. Como ya se ha mencionado en el Apartado 7.2, las redes neuronales tradicionales, como es el caso del MLP cuentas con 3 capas diferenciadas. Las neuronas de la capa de entrada no realizan el mismo cómputo que el resto, es decir, no actúan como neuronas propiamente dichas. Simplemente recogen la entrada (señales) y se encargan de propagarlas a todas las neuronas de la siguiente capa. Las capas ocultas se encargan del procesamiento de las entradas, y la capa de salida es la encargada de proporcionar al exterior la respuesta de la red. A continuación se muestra un esquema de la arquitectura de un perceptrón multicapa –Figura 8-.

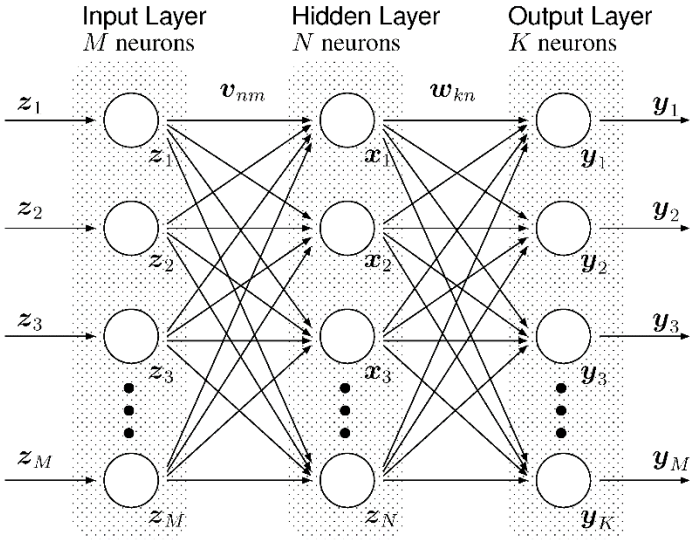

Figura 8. Perceptrón multicapa [14]

Como se puede observar, las conexiones del Perceptrón Multicapa están dirigidas hacia delante. Por ese motivo este tipo de redes se suelen denominar redes "alimentadas hacia delante" o feedforward. Normalmente los MLP están completamente conectados. Esto quiere decir que todas las neuronas de una capa están conectadas con todas las neuronas de la capa siguiente. Cuando se abordan problemas de clasificación se suele partir de este tipo de arquitectura totalmente conectada.

A la hora de realizar el diseño de un perceptrón multicapa hay algunas consideraciones a tener en cuenta. El número de capas y de neuronas tomará cierta importancia a la hora de la obtención mejores o peores resultados. Otra de las elecciones más importantes será la de la función de activación, las cuales también serán comentadas. Dicha elección será elegida por el diseñador de la red, y se hará en base a los valores de la activación que se desean alcanzar en las neuronas. La fórmula que relaciona ambas funciones es fthip(x) = 2fsigm(x) – 1, por lo tanto, la elección de una u otra función iría en base al recorrido que mas interese.

9.1.2. Propagación de patrones de entrada

El perceptrón multicapa define una relación entre variables de entrada y salida. Esta relación la obtenemos mediante la propagación hacia delante de los procesamientos de datos de cada una de las neuronas hacia las siguientes capas. Dependiendo de las entradas que recibe cada neurona, se produce una activación que se propaga recorriendo la arquitectura de la red. Hay diferentes

ecuaciones para el cálculo de las activaciones que toman parte en el proceso de propagación de patrones:

Activación de la capa de entrada:

Las activaciones de las neuronas de la capa de entrada son básicamente las entradas de la red, las cuales serán pasadas a las siguientes neuronas:

$$a1i = xi$$

$$i = 1,2,\ldots,n$$

Activación de las capas ocultas:

Las activaciones vienen dadas por la aplicación de la función de activación al procesado de información que les llega de la capa anterior. Este procesamiento se basa en los productos de las activaciones por sus correspondientes pesos:

$$a_i^c = \mathfrak{f}\left(\sum_{j=1}^{n_{c=1}} w_{ji}^{c-1}\, a_j^{c-1} + u_i^c\right)$$

$$i = 1,2,\ldots,n_c$$

$$c = 2,3,\ldots c-1$$

$$donde\ a_j^{c-1}\ son\ las\ activaciones\ de\ las\ neuronas\ de\ la\ capa\ c-1$$

Activación de la capa de salida:

$$y_i = a_i^c = \mathfrak{f}\left(\sum_{j=1}^{n_{c=1}} w_{ji}^{c-1}\, a_j^{c-1} + u_i^c\right)$$

$$i = 1,2,\ldots,n_c$$

$$donde\ Y = (y1, y2, \ldots, y_{nc})$$

La función ƒ que se aplica en estas formulas son denominadas como función de activación. Hay un gran número de funciones de activación que se pueden usar en las redes neuronales. Sin

embargo, hay ciertas funciones más apropiadas para determinados dominios y problemas a resolver. Algunas de las funciones existentes y más utilizadas son las siguientes:

Función sigmoidal:

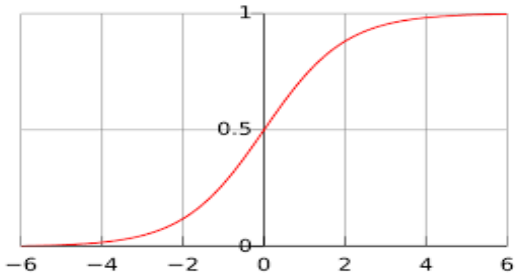

Figura 9. Función sigmoidal

$$f_{asing}(x) = \frac{1}{1 + e^{-x}}$$

Función tangente hiperbólica:

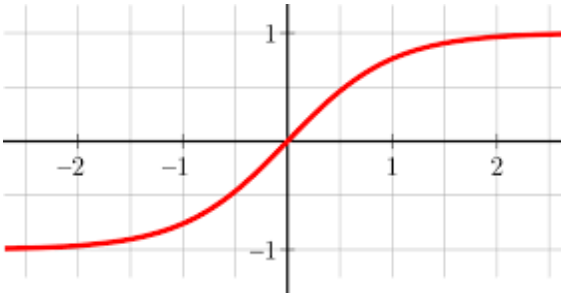

Figura 10. Función tangente hiperbólica

$$f_{thip}(x) = \frac{1 - e^{-x}}{1 + e^{-x}}$$

Dichas funciones poseen como imagen intervalos continuos de valores [0,1] para la función sigmoidal y [-1,1] para la tangente hiperbólica.

Rectified Linear Units (ReLUs):

En lugar de funciones del tipo anterior, actualmente en redes de aprendizaje más profundo se están utilizando funciones de tipo ReLU para las capas ocultas. Estas funciones tienen como salida 0 si la entrada es menor que 0, y la entrada sin procesar en caso contrario, es decir, si la

entrada es mayor que 0, la salida de la neurona será igual que la entrada. Las funciones de ReLU son una de las funciones más parecidas a la función que en teoría usan las neuronas de nuestro cerebro. Algunas investigaciones (Toronto [15]) muestran que estas funciones son mucho más rápidas que el resto con largos entrenamientos de redes. Sin embargo, aun siendo una función muy simple y que funciona muy bien no quiere decir que sea perfecta. En ocasiones el gradiente puede afectar a las neuronas con este tipo de funciones de tal modo que los pesos se actualicen de tal forma que nunca más esta neurona se vuelva a activar. Si eso ocurre el gradiente será 0 para cualquier momento siguiente en el entrenamiento, y las neuronas con esta función estarán "muertas" para el resto del proceso de aprendizaje.

$$f(x) = max\ (x, 0)$$

Softmax:

Esta función es usada principalmente con problemas de clasificación y se relaciona con las capas de salida de la red . La función softmax [16] hace que cada salida este en un valor comprendido entre 0 y 1, partiendo de la base de que la suma total de todas esas salidas es igual a 1. Por lo tanto, se puede ver que la salida de la función softmax es equivalente a una distribución de probabilidad categórica, dándote la probabilidad de cada clase de ser verdadera, lo que funciona muy bien en clasificaciones.

$$\sigma(z)_i = \frac{e^{zj}}{\sum_{k=1}^{K} e^{zk}}$$

Esta ecuación arrojaría una serie de probabilidades, como por ejemplo las siguientes, las cuales se muestran en forma de histograma –Figura 11-. Este ejemplo en concreto correspondería a la salida de la red neuronal como respuesta a una entrada del dominio MNIST –Apartado 11.1-, la cual seguramente se corresponda con el número manuscrito 4

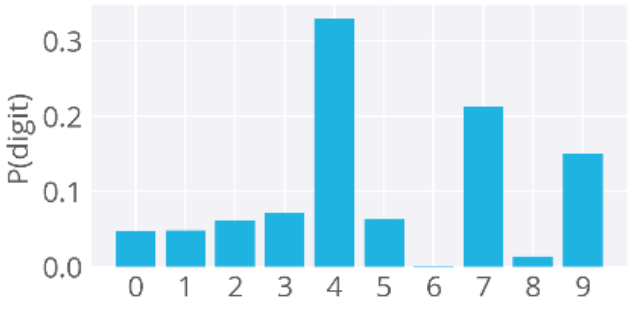

Figura 11. Salida de la función softmax [17]

En numerosas ocasiones, la función de activación del MLP será común a todas las neuronas de la red. Sin embargo, y dependiendo de la naturaleza del problema que se desea abordar, las neuronas de la capa de salida se pueden distinguir del resto de neuronas con la utilización de otro tipo de función de activación. Uno de los ejemplos es la función Softmax. Esta función estará ubicada únicamente en la capa de salida, mientras que en las neuronas de las capas ocultas podemos tener una función de activación diferente como la ReLU o la tangente hiperbólica.

9.1.3. Algoritmo de retropropagación

Estos algoritmos (algoritmo de aprendizaje) son los encargados de la modificación de parámetros de la red y de la adaptación de la misma. Cuando trabajamos con MLP´s este algoritmo se basará en un aprendizaje supervisado, debido a que la modificación de los parámetros se realiza con el fin de que la salida de la red sea lo más próxima a la salida esperada.

Como en todo algoritmo supervisado es necesario contar con un patrón de salida para cada uno de los patrones de entrada. Sabiendo que lo que se quiere obtener es una salida lo más parecida posible a esta salida deseada, el problema se puede formular como un problema de minimización:

$$Min_W E$$

W será el conjunto de parámetros de la red y E es una función de Error que mide la diferencia entre la salida de la red y las salidas deseadas. Este error será calculado de la siguiente forma:

$$E = \frac{1}{N}\sum_{n=1}^{N} e(n) \rightarrow e(n) = \frac{1}{n_C}\sum_{i=1}^{n_C}\left(s_i(n) - y_i(n)\right)^2$$

Siendo N el número de patrones a analizar y e(n) es el error cometido por la red para el patrón n, teniendo a Y(n) como los vectores de salidas obtenidas y S(n) como las salidas deseadas por la red para el patrón n.

Por lo tanto, este proceso es equivalente a la búsqueda de un mínimo en la función de error. Sin embargo, el uso de funciones de activación no lineales hace que la respuesta de la red sea no lineal en base a los parámetros ajustables. Por lo tanto, será necesaria la aplicación de técnicas de

optimización no lineales para su resolución siguiendo una dirección de búsqueda. En el contexto de las redes neuronales, y más en concreto con el perceptrón multicapa, la dirección de búsqueda más usada es la dirección negativa del gradiente en función de E, el denominado método del descenso del gradiente, debido a que se sabe que es la dirección en la que la función decrece.

Una parte muy importante a tener en cuenta en las redes neuronales es que la minimización del error no se hace sobre el error total de la red, sino que consiste en el cálculo sucesivo de errores de cada patrón de entrada e(n). De este modo cada uno de los parámetros de la red se modifica para cada patrón de entrada:

$$w(\mathrm{n}) = \mathrm{w}(\mathrm{n}-1) - \alpha \frac{\partial_{\mathrm{e}}(\mathrm{n})}{\partial_{\mathrm{w}}}$$

Siendo α la razón de aprendizaje de la red. Lo que realmente representa dicha tasa es el "desplazamiento" que se produce en el espacio de búsqueda. Cuanto mayor sea esta tasa de aprendizaje, mayores serán los saltos que se produzcan a lo largo del espacio. ¿Qué puede implicar esto? Si reducimos mucho la tasa de aprendizaje, es posible que, al realizar pequeños desplazamientos por ese espacio de búsqueda, se caiga en un mínimo local que no represente una solución óptima. Aplicando el método del descenso de gradiente de una forma eficiente, debido a la estructuración en capas de las redes neuronales, se desarrolla el algoritmo de retropropagación, también conocido como la regla delta generalizada.

9.1.4. Regla delta generalizada

En este apartado se va a proceder a una explicación simplificada de los cálculos necesarios del algoritmo de retropropagación. Lo que se basa el aprendizaje de una red neuronal es en la adaptación de los pesos siguiendo una dirección de búsqueda. En este caso la dirección que se sigue es la dirección negativa del gradiente en función del error.

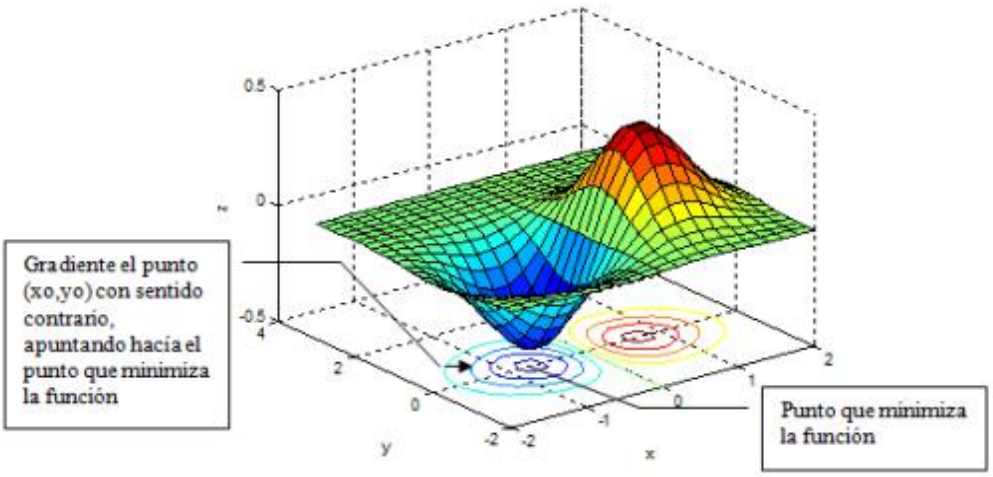

Figura 12. Descenso de gradiente

Uno de los grandes problemas del MLP es la existencia de mínimos locales en el espacio de búsqueda. Si en el proceso de aprendizaje se acaba cayendo en un mínimo local, la red neuronal se puede quedar estancada, sin ser este punto el que minimiza el error de manera óptima.

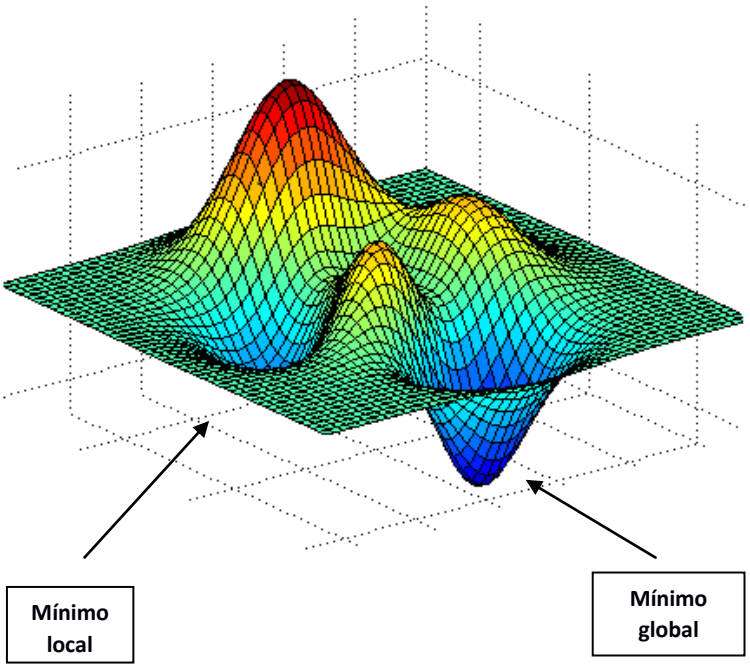

Figura 13. Mínimo local vs Mínimo global

Hay que distinguir dos casos en los cuales los cálculos son diferentes. Por un lado tenemos la aplicación del algoritmo para los pesos de la capa oculta conectada a la capa de salida, y por otro lado tenemos el resto de pesos y umbrales de la red.

Actualización de pesos de la capa oculta C-1 con la capa y umbrales de salida:

Utilizado la formula anteriormente explicada, la modificación del peso de la capa c-1 de la neurona j con la neurona i de la capa de salida seguirá la siguiente fórmula:

$$w_{ji}^{C-1}(n) = w_{ji}^{C-1}(n-1) - \alpha \frac{\partial_{e(n)}}{\partial w_{ji}^{C-1}}$$

Como se puede comprobar, es necesaria la evaluación del error e(n) para dicho punto. SI aplicamos ambas formulas anteriormente explicadas:

$$\frac{\partial_{e(n)}}{\partial w_{ji}^{C-1}} = -\left(s_i(n) - y_i(n)\right)\frac{\partial y_i(n)}{\partial w_{ji}^{C-1}} \rightarrow \frac{\partial y_i(n)}{\partial w_{ji}^{C-1}} = f'\left(\sum_{j=1}^{nC-1} w_{ji}^{C-1}\, a_j^{C-1} + u_i^C\right) a_j^{C-1}(n)$$

Es necesario calcular la derivada de la neurona de salida respecto del peso. La salida se corresponde con la función de activación que es aplicada al sumatorio de los productos de los pesos por las entradas más los umbrales. Con ello se obtiene el término δ_i^C. Este valor va asociado a la neurona i de la capa C y al patrón n.

$$\delta_i^C = -\left(s_i(n) - y_i(n)\right) f'\left(\sum_{j=1}^{nC-1} w_{ji}^{C-1}\, a_j^{C-1} + u_i^C\right) \rightarrow \frac{\partial_{e(n)}}{\partial w_{ji}^{C-1}} = \delta_i^C(n) a_j^{C-1}(n)$$

Si se sustituye la derivada del error e(n) respecto al peso w_{ji}^{C-1} , se obtiene finalmente la ecuación para la modificación del peso.

$$w_{ji}^{C-1}(n) = w_{ji}^{C-1}(n-1) + \alpha\delta_i^C(n)a_j^{C-1}(n)$$

Para los umbrales simplemente hay que realizar una pequeña modificación. Como ya se ha explicado, en un MLP el umbral es otra conexión, la cual adquiere un valor constante de entrada.

$$u_i^C(n) = u_i^C(n-1) + \alpha\delta_i^C(n)$$

La conclusión que podemos extraer es que en la modificación de un peso que une la neurona j con una neurona i de la capa de salida, basta con tener en cuenta la activación de la neurona de la que viene la conexión.

Actualización de los pesos y umbrales del resto de capas:

$$w_{ji}^{C-1}(n) = w_{ji}^{C-1}(n-1) + \alpha\frac{\partial_{e(n)}}{\partial w_{ji}^{C-1}}$$

En base a los cálculos realizados en la actualización de los pesos y umbrales de la penúltima y la última capa, muchos de los cálculos son extrapolables. Sabiendo que el término δ_j^{C+1} en este caso es obtenido de la siguiente manera:

$$\delta_j^{C+1} = f'\left(\sum_{k=1}^{nc} w_{kj}^c a_k^c + u_j^{c+1}\right)\sum_{i=1}^{nc+2}\delta_i^{c+2}w_{ji}^{c+1} \rightarrow \delta_j^{C+1} = a_j^{c+1}(1-a_j^{c+1})\sum_{i=1}^{nc+2}\delta_i^{c+2}w_{ji}^{c+1}$$

Conociendo el valor que adquiere el término δ_j^{C+1} se puede obtener el valor de pesos y umbrales en capas intermedias:

$$w_{kj}^{C}(n) = w_{kj}^{C}(n-1) + \alpha\delta_i^{C+1}(n)a_k^{C}(n)$$

$$u_j^{C+1}(n) = u_j^{C+1}(n-1) + \alpha\delta_j^{C+1}(n)$$

9.2. Redes **convolucionales** (CNN)

Las redes neuronales convolucionales son actualmente unas de las redes neuronales más usadas a la hora de la clasificación de imágenes. Esta clase de red del tipo feedforward, se puede considerar como una variación del perceptrón multicapa. Está inspirada en procesos biológicos donde el patrón de conectividad entre las neuronas se asemeja a la organización de la corteza visual animal. Estas neuronas corticales responden a los estímulos solo en una región restringida del campo visual conocido como el campo receptivo. Estos campos receptivos a su vez se superponen parcialmente de tal modo que cubren finalmente todo el campo visual.

Lo que hace a estas redes muy convenientes para el análisis de imágenes es que realizan poco preprocesamiento en comparación con otros algoritmos de clasificación de imágenes. Estas redes aprenden los filtros que, algoritmos más tradicionales, los obtienen ya que fueron diseñados a mano. Las redes neuronales clásicas, como el MLP, en teoría no son capaces de adatarse bien a las imágenes, a diferencia de este tipo de redes, las cuales tienen un diseño de arquitectura específico para el tratamiento de imágenes. En particular, a diferencia de las redes neuronales clásicas, las capas de una CNN tienen neuronas dispuestas en 3 dimensiones: alto, ancho y profundo. En el ejemplo del CIFAR 10 que se mostrará más adelante, las imágenes de este dominio tienen dimensiones de 32x32x3. Es decir, son imágenes de 32x32 con 3 capas de color.

La peculiaridad, y lo que hace a estas redes neuronales más eficientes frente a los MLP, es que las neuronas de una capa solo estarán conectadas a una pequeña región de la capa anterior, en lugar de estar completamente conectadas. La completa conectividad de los MLP puede llegar a ser un desperdicio: Hay que tener en cuenta también que la gran cantidad de parámetros puede llevar a estas redes a caer en un rápido sobreajuste.

9.2.1. Arquitectura

Las CNN suelen seguir un patrón a la hora de la organización de su arquitectura. Estas redes neuronales cuentan con una capa convolucional (**Convolutional Layer**), otra capa denominada **Pooling Layer**, y finalmente una capa completamente conectada (Fully-connected layer) la cual se corresponde con una red neuronal clásica, un MLP.

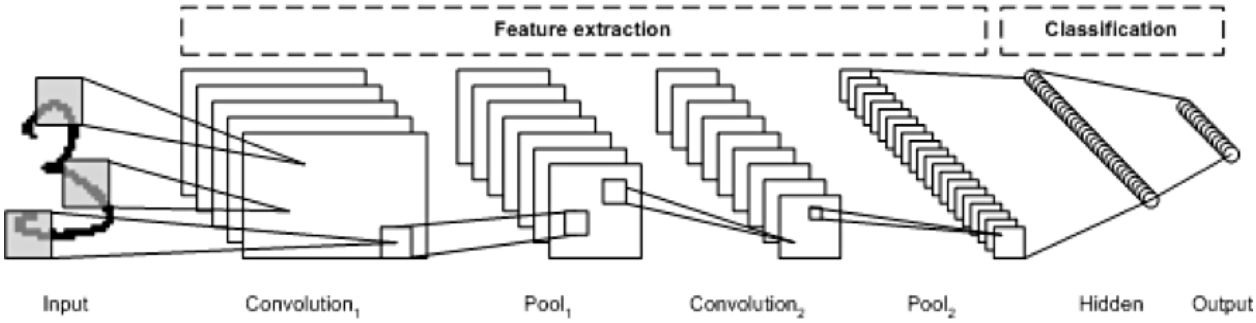

Figura 14. Red convolucional [18]

9.2.2. Capa convolucional (Convolutional layer)

Esta capa se podría decir que es el núcleo y parte más importante en una red convolucional, y es su capa más característica y en la que mas difiere con un MLP, por lo tanto, es interesante profundizar en su funcionamiento.

Cuando se trabaja con dominios en los cuales las entradas tienen una dimensionalidad enorme, se suele aplicar una conectividad local. Es decir, se conecta cada neurona con una región en particular de todo el volumen de entrada. La representación espacial de esta conectividad es un hiperplano denominado “campo receptivo de la neurona” (receptive field of the neuron).

Ejemplo: Imaginemos que tenemos una imagen de entrada del dominio CIfar10 (32x32x3) y contamos con un campo neuronal receptivo de 5x5. Si esto es así, cada neurona tendrá asociada una región de pesos de 5x5x3. Destacar que mientras que la altura y la anchura de la imagen de entrada no coincide con las del campo receptivo, su profundidad es 3, debido a que esta es la profundidad del volumen de entrada.

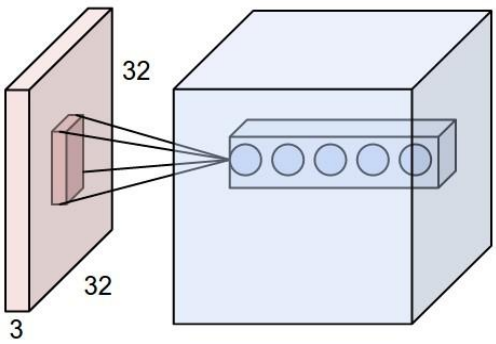

Figura 15.Convolución

Esto explicaría la conectividad de cada una de las neuronas en la capa convolucional para un cierto volumen de entrada, pero no las neuronas de salida para dicho volumen. Habría tres parámetros de control para el volumen de salida: la profundidad (depth), el desplazamiento (stride) y el rellenado de zeros (zero-padding)

Por un lado la profundidad se corresponde con el número de filtros que se quieren usar, obteniendo con cada uno de estos una característica diferente en la imagen de entrada. El desplazamiento hace referencia al movimiento del filtro. Si aumentamos el desplazamiento de esta matriz, el volumen de salida será más pequeño. En ciertas ocasiones será necesario realizar un rellenado de ceros en el borde de la imagen, con el fin de poder preservar exactamente el volumen de entrada para que el ancho y alto de entrada y salida sean el mismo.

Sabiendo un cierto volumen de entrada, y estos 3 parámetros, se puede calcular el número de neuronas que se generarán en el espacio de salida. Eso es importante a la hora de generar una arquitectura de red neuronal, para controlar el volumen de las capas y el número de neuronas necesarias. A continuación se pasa a presentar un pequeño ejemplo para el entendimiento de estos cálculos:

El volumen de la entrada será representado por W, la región de neuronas localmente conectada vendrá denotada por F, el desplazamiento de la región de neuronas será S, y finalmente el rellenado de 0's será P. La formula a utilizar es la siguiente:

$$((W - F + 2P)/S + 1$$

Si partimos de un ejemplo en el que tengamos un volumen de entrada de 7x7, y la región localmente conectada de 3x3 con un desplazamiento de una unidad sin ningún tipo de rellenado de 0's, se obtiene un conjunto de 5x5 neuronas, cada una de ellas con 3x3 entradas.

En muchas ocasiones el número de neuronas necesarias será inmenso, y será muy conveniente realizar un procedimiento denominado intercambio de parámetros o **Parameter Sharing.** Para explicar esto se va a pasar a presentar un ejemplo del mundo real, en el cual se utilizó una red neuronal ganadora de la competición ImageNet en 2012**.** Esta red, denominada arquitectura de Krizhevsky, era capaz de recibir como entrada imágenes de entrada de 227x227x3. Decidieron que su primera capa convolucional tendría un tamaño de 11x11 con un desplazamiento de 4 y sin ningún rellenado de ceros. Aplicando la formula se obtiene lo siguiente: ((227 – 11)/4)+1 = 55.

La profundidad de la capa convolucional realiza la extracción de una característica concreta de la imagen. Definieron una profundidad de 96 en esta capa convolucional, por lo tanto, el volumen de salida de esta capa sería 55x55x96, en la que cada neurona tendría como entrada un total de

11x11x3 pesos del volumen total de entrada, lo cual como se verá más adelante es una cantidad ingente de parámetros a tener en cuenta.

9.2.3. Intercambio de parámetros

Para controlar el abundante numero de parámetros que se generan al trabajar en dominios con imágenes, es frecuente el uso de intercambio de parámetros en las capas convolucionales. Si se utiliza el ejemplo anteriormente visto, se tendrían 55 x 55 x 96 = 290400 neuronas en la primera capa, cada una de ellas con 11 x 11 x 3 = 363 ponderaciones y un sesgo. Todo ellos haría un total de 105705600 parámetros, y eso tan solo para la primera capa convolucional.

Sin embargo, se puede reducir drásticamente este número mediante una pequeña suposición: si una característica es útil para calcular en alguna posición espacial, por ejemplo x,y , entonces esa característica también debería ser útil para calcular en otra posición diferente x2, y2. Dicho de otra manera, partiendo del mismo ejemplo anteriormente citado, en el que tenemos un volumen en la capa convolucional de 55x55x96, tendríamos 96 cortes en profundidad, cada uno de ellos de un tamaño de 55x55. Es decir, se restringe a que en cada segmento de profundidad se usen los mismos pesos y sesgos. La primera capa convolucional por lo tanto contaría con 96 conjuntos únicos de ponderaciones (uno para cada segmento de profundidad), teniendo así un total de 96x11x11x3 = 34848 ponderaciones únicas o parámetros, más los 96 sesgos. Por otro lado, las 55x55 neuronas que habría en cada sector de profundidad tendrían el mismo vector de pesos.

(Ejemplo obtenido de la referencia [19])

Teniendo esto en cuenta, el pase de dichos pesos hacia adelante puede calcularse en cada segmento de profundidad como una sola convolución de los pesos de la neurona con el volumen de entrada.

Sin embargo, hay ocasiones en las que no tiene sentido este intercambio de parámetros. Esto ocurre especialmente cuando necesitamos, para un dominio dado, aprendes características completamente diferentes en un lado de la imagen y en otro. Por ejemplo, se puede ver esto con más claridad cuando nos referimos a un dominio en el que las entradas son imágenes de rostros. Es de esperar que se tengan que aprender características específicas del ojo y de la boca en diferentes ubicaciones espaciales. En este caso sería conveniente relajar el esquema de intercambio de parámetros, y simplemente llamar a la capa localmente conectada.

9.2.4. Capa Pooling (Pooling layer)

Esta es una capa que se utiliza principalmente para la reducción progresiva del tamaño espacial con el que estamos trabajando, para poder disminuir la cantidad de parámetros y cálculos en la red y, por lo tanto, intentar controlar el sobreajuste.

Este tipo de capas opera independientemente en cada sector de profundidad de la entrada, reduciendo su tamaño mediante la operación MAX. Lo que suele ser más común en las capas de pooling es el uso de un kernel de tamaño 2x2 con un desplazamiento de 2 pixeles. Con esto se puede descartar el 75% de las activaciones. Cada uno de estos kernels recogería una región cuadrada de 4 números, (en este caso específico) devolviendo el mayor de todos ellos. La dimensión de profundidad de la entrada se mantiene invariante.

La función que utiliza la capa de pooling no tiene por qué ser la de MAX. Se pueden usar otras variaciones como la media o el "spatial pyramid pooling". A lo largo de los últimos años se utilizaba con más frecuencia la media en este tipo de capas, sin embargo, con el paso del tiempo la función Max es la que ha ido cogiendo más fuerza, y se considera que es la que mejor funciona en la gran mayoría de los casos.

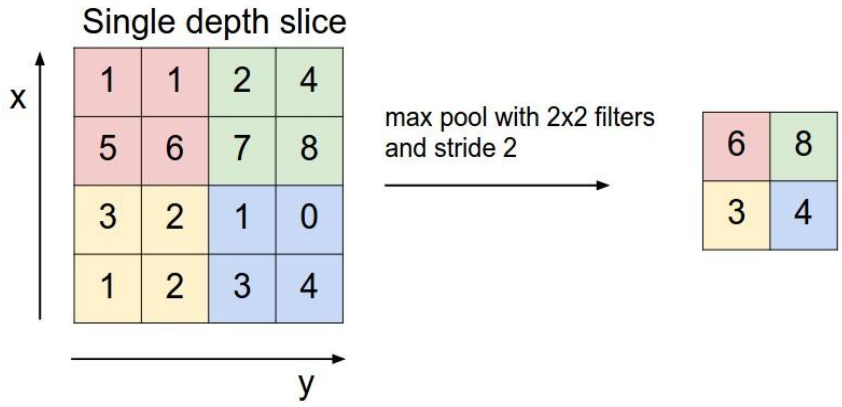

Figura 16. Función Max-Pool

Como se ha comentado anteriormente, en este caso en el que usamos un kernel para el pooling de 2x2, conseguimos reducir la imagen un 75%; y como se puede ver en la siguiente imagen, esta reducción no afecta al volumen de profundidad de la misma:

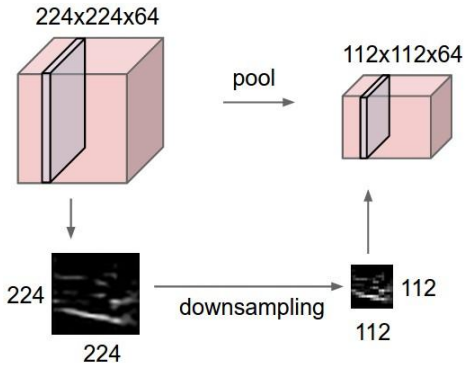

Figura 17. Reducción Max-Pool

9.2.5. Capa completamente conectada

Como ya se ha mencionado, la ultima capa de las redes convolucionales es una red neuronal clásica (MLP), es decir, una serie de neuronas completamente conectadas. Por lo tanto, sus activaciones se pueden calcular como la multiplicación de matrices seguidas de la suma del sesgo. El número de estas capas al final de una red convolucional puede variar, al igual que lo hace en un MLP. Por otro lado en cuanto al algoritmo de retropropagación se refieres y la regla delta generalizada, mostrada en el apartado 9.5 del MLP, sus cálculos son idénticos y extrapolables.

10. DOMINIOS PARA LA CLASIFICACIÓN DE IMÁGENES

El crecimiento exponencial de datos a analizar ha hecho que, en los últimos años, se aumente considerablemente el número de datasets de los que disponer para poder extraer información.

10.1. MNIST

EL dominio MNIST [20] es un dataset el cual contiene un gran número de dígitos manuscritos. Es uno de los dominios con los que se debería comenzar si uno se inicia en el mundo de tratamiento y reconocimiento de caracteres, debido a que es uno de los más sencillos y sobre el que prácticamente más información y trabajos relacionados se pueden encontrar.

El dataset del MNIST consiste en un total de 60000 imágenes de cada uno de los 10 dígitos existentes en el sistema decimal. Este conjunto es el usado como conjunto de entrenamiento. Por otro lado se cuentan con otras 10000 imágenes del mismo tipo, sin embargo estas serán usadas como conjunto de test, es decir, para evaluar como de bueno es el modelo generado con el conjunto de entrenamiento. Las imágenes en este dominio tienen unas dimensiones de 28x28 = 784 pixeles en escala de grises. Cada una de las imágenes irá asociada a una etiqueta la cual representa su respectiva clase, las cuales irá, de 0 a 9:

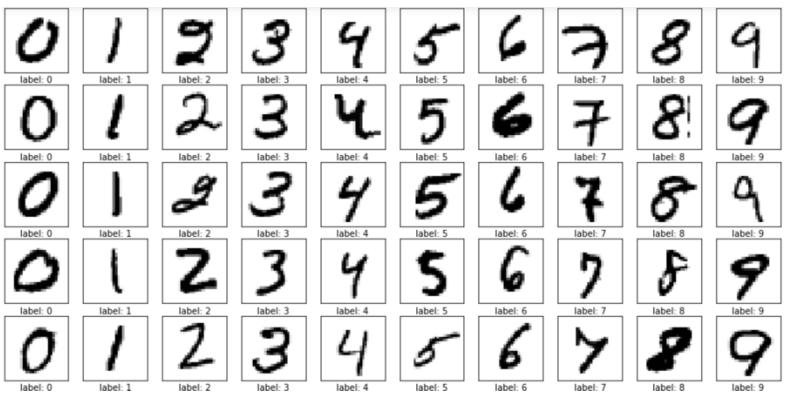

Figura 18. Dominio MNIST

10.2. CIFAR10

CIFAR10 [21] es un conjunto de datos a analizar es en realidad un subconjunto de un dominio más grande: el CIFAR100. Cada una de las imágenes que componen el dominio tienen 32x32x3 = 3072 pixeles.

A diferencia del MNIST, aquí las imágenes no son en escala de grises, sino que vienen en un formato con 3 capas de color, de ahí la tercera dimensión añadida en cada una de las imágenes.

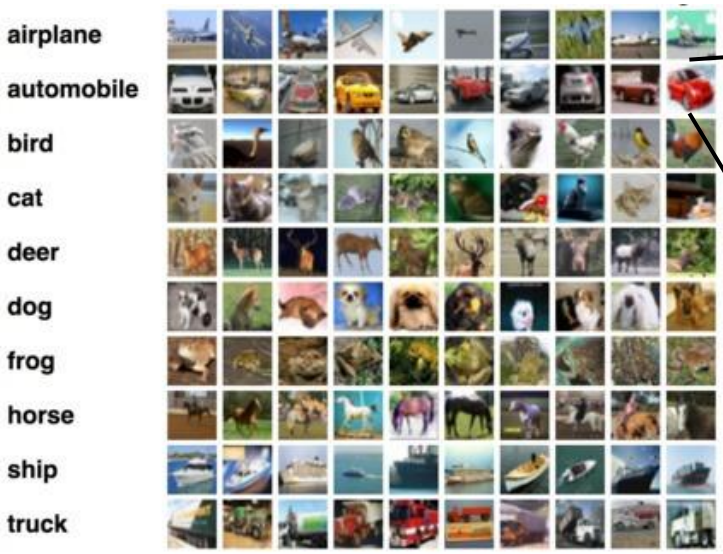

Figura 19. Dominio CIFAR10

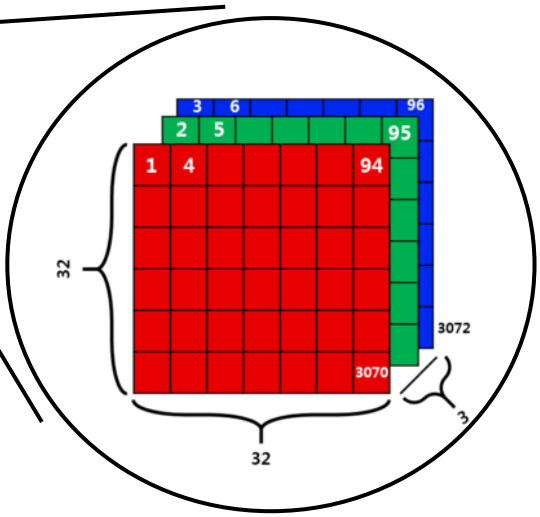

Figura 20. Estructura imagen CIFAR10

EL CIFAR-10 es un dataset que consta de 60000 imágenes, de las cuales 50000 son para el conjunto de entrenamiento y las otras 10000 para el conjunto de test. Todo el conjunto está dividido en “batches” de 10000 imágenes. Los 5 batches del conjunto de entrenamiento están aleatorizados. Es decir, las imágenes que hay en cada batch están ordenadas aleatoriamente. De este modo, se intenta generar un modelo que generalice todo lo posible. Aunque estos batches de entrenamiento estén aleatorizados, no se garantiza que haya más clases de imágenes que otras en cada batch, aunque no tiene porque ser un problema significativo. Por otro lado y a diferencia de los conjuntos de entrenamiento, el conjunto de test esta mucho mas balanceado. Contiene 10 grupos de 1000 imágenes aleatorizadas de cada una de las clases. Así se asegura que al pasar el conjunto de test se valide con el mismo número de imágenes de cada una de las clases.

Cabe destacar que en ese dominio las clases son mutuamente exclusivas. Esto quiere decir que, por ejemplo, no hay solapamiento entre la clase coche y la clase camión.

11. LENGUAJE UTILIZADO

La elección del lenguaje de programación Python viene dada ya que es uno de los lenguajes más versátiles y que más facilidades ofrece a la hora de realizar tareas de machine learning. Frente al resto de lenguajes de programación, python ofrece más facilidad de uso, una enorme base de usuarios, y numerosas bibliotecas destinadas al deep learning. Hay que destacar que Python no es un lenguaje realmente optimizado, como pueda ser C. Sin embargo, el uso conjunto de las librerías de tensorflow y numpy hace que los procesos de machine learning se optimicen algo más.

¿Por qué el uso de tensorflow?

Gracias a tensorflow se pueden definir cálculos matemáticos (La base de toda red neuronal) mediante un grafo de nodos y arcos –Figura 21-. En este tipo de grafos, los nodos representan operaciones matemáticas o módulos de cálculo dentro del código. En ocasiones también pueden representar puntos para la "alimentación de datos", entrada/salida etc.

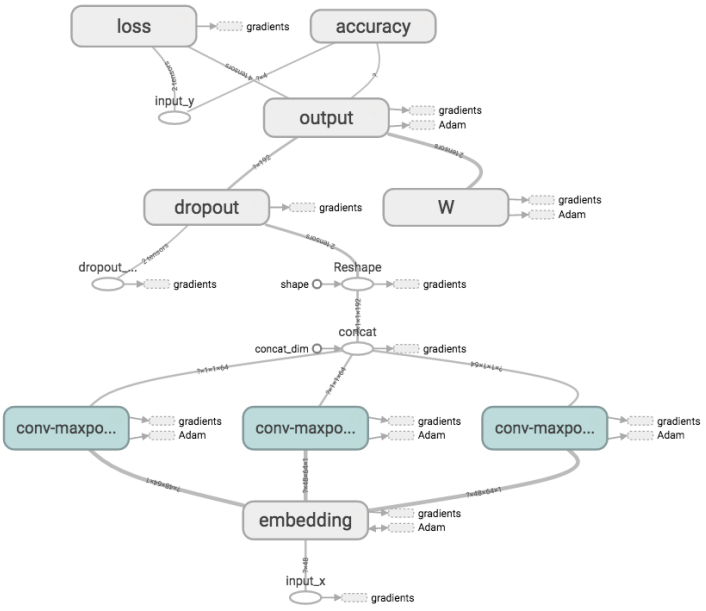

Figura 21. Grafo generado por TensorFlow

Los arcos del grafo representan las relaciones existentes entre estos módulos de cálculo, las cuales definen arreglos de datos multidimensionales, también denominados tensores.

Todo ello genera un grafo, el cual representa la red neuronal que se ha generado. Gracias a un servicio que ofrece tensorflow denominado tensorboard, se pueden visualizar estos grafos, los cuales en numerosas ocasiones requieren de una ardua tarea de estudio y entendimiento. Es importante entender que los nodos (dispositivos computacionales) comienzan a ejecutarse, de manera asíncrona y en paralelo, una vez que todos los tensores de entrada están disponibles.

¿Qué es TensorBoard?

Es un servicio que nos proporciona Tensorflow, el cual nos permite visualizar el cambio de variables y tensores a lo largo de la ejecución de un entrenamiento de una red neuronal. Esto lo hace mediante la lectura de archivos eventuales generados por los tensores, los cuales contienen datos de resumen (summary data) que tensorflow genera a lo largo de su ejecución. Gracias a Tensorboard, es más fácil entender, debuguear, t optimizar programas de Tensorflow. Como ya se ha explicado antes, también se puede visualizar el grafo que representa la red neuronal.

12. ELECCIÓN DE LA ARQUITECTURA INICIAL

12.1. MLP:

Lo más importante en todo entrenamiento de un modelo son los datos. Son parte imprescindible, y por ellos, la arquitectura se ha de adaptar a ellos lo máximo posible para poder explotarlos de manera más efectiva. Por ello hay que realizar una serie de comprobaciones previas a la elección de la arquitectura.

Como bien se ha explicado en el apartado de los dominios, el MNIST consta de imágenes de 28x28 pixeles, cada uno de ellos codificado como un número. Por lo tanto, cada imagen representa una entrada de 784. Esto quiere decir que en la primera capa oculta del perceptrón multicapa se tendrán X neuronas (Siendo x el numero que se decida más adelante) cada una de ellas con 784 entradas. En cuanto al CIFAR10 se refiere, la arquitectura constará del mismo número de neuronas de salida que con el MNIST, sin embargo, en la capa de entrada difiere del anterior. Conociendo la disposición de los datos en el dominio, se requiere de 32x32x3 = 3072 neuronas.

El tema de la elección del número de neuronas es algo muy discutido. No parece que haya una regla universal y exacta para tomar esta decisión. Sin embargo es una de las decisiones más importantes a tomar a la hora de elegir un modelo de arquitectura.

Como ya se ha explicado anteriormente, el MLP tiene 3 zonas diferenciadas: Capa de entrada, capa/s oculta/s y capa de salida. El numero de neuronas tanto la capa de entrada como la de salida están más condicionadas debido a que tienen una interacción directa con el exterior. La elección difícil viene de la mano de la capa/s oculta/s.

Para la capa de entrada se sabe, como ya se ha comentado, que tendremos 784 entradas, y para la capa de salida un total de 10 neuronas, una por cada posible clase dentro del dominio.

Ahora bien, tenemos diferentes métodos para la elección de las neuronas ocultas.

Método de prueba y error:

Es uno de los métodos más difundidos, y consiste básicamente, en probar diferentes números de neuronas y capas, y en base a los resultados obtenidos, ir cambiado este numero de neuronas para ir mejorando dichos resultados. Existen dos variantes que parecen lógicas en este método, las cuales son el “Forward Approach”, basado en la ejecución del modelo partiendo de un

número muy bajo de neuronas, e ir incrementando dicho numero hasta alcanzar unas tasas de precisión correctas; y el "Backward Approach" en el que se realizaría el proceso totalmente contrario.

Método "Thumb":

Este método tiene varias versiones, pero tiene dos más destacadas. Una de ellas es que el número de neuronas ocultas en cada capa tiene que estar entre el tamaño de entrada de la red y el de salida. Es decir, cada capa no debería tener más de 784 neuronas ni menos de 10. La otra dice que el numero de neuronas ocultas debería ser 2/3 del tamaño de entrada más el tamaño de salida, es decir, unas 530 neuronas ocultas. Ambas versiones son compatibles y se podrían usar como punto de partida de la red, y a partir de ahí realizar el proceso de prueba y error hacia delante y hacia atrás.

Estos métodos nos pueden dar una idea acerca del número de neuronas adecuado dentro de la red. Sin embargo, el número de capas ocultas es un tema algo más complejo, en el que podemos encontrar dos tipos de opiniones.

La primera de ellas es la que se basa en el Teorema de Aproximación Universal [22]. Este teorema se aplica a las RNA y establece que una red neuronal es capaz de aproximar cualquier función continua con un número finito de discontinuidades teniendo en cuenta que se use como función de activación una no continua. Según un artículo publicado por Funahashi, una función continua definida en un conjunto compacto Rn puede ser aproximada por una red neuronal que cuenta con una capa oculta [22].

Por otro lado, está demostrado que, el aumento del número de capas ocultas dentro de una red neuronal añade dimensionalidad a la misma. A medida que se aumenta la dimensionalidad, se pueden resolver problemas más complejos, como se puede observar en el ejemplo de la función XOR en el apartado 8.1. Inicialmente se optará por una capa oculta, y posteriormente se irá aumentado el número de capas.

El proceso de elección del resto de parámetros de la red neuronal es lógico elegirlo en base a procedimientos de prueba y error, aunque conociendo el funcionamiento teórico podemos realizar aproximaciones, por ejemplo, de la elección de funciones de activación.

Funciones del tipo sigmoide:

Ha sido una función de activación muy usada a lo largo de los años en machine learning, Sin embargo, con el paso del tiempo ha caído en decadencia, sobre todo en este tipo de dominios en los que se requiere un gran número de neuronas y varias capas ocultas.

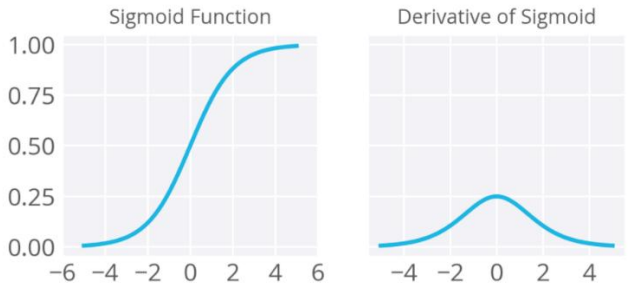

Figura 22. Derivada de la función sigmoidal

Si nos fijamos en la derivada de dicha función -Figura 22- podremos apreciar que, en el proceso de backpropagation, el error se puede ver reducido en una cuarta parte de lo que en realidad era. Cuanto más se avance en las sucesivas capas de la red, mas conocimientos de los datos se irá perdiendo. De este modo, ciertos errores en las capas más profundas no afectarán a neuronas de capas poco profundas.

ReLU:

Se sabe que para este dominio, es recomendable el uso de funciones de activación de tipo ReLU. Sin embargo, si se tienen grandes descensos de gradiente, estos pueden afectar a las neuronas con este tipo de activación, y por consiguiente "matarlas". Esto ocurre debido a una actualización de los pesos (causada por dicho gradiente) que imposibilita la activación de esa neurona en el resto del entrenamiento, debido a que el descenso de gradiente en ese punto será 0, haciendo que esta quede inservible. Esto puede ocurrirle a numerosas neuronas dentro de la red, ocasionando pérdidas importantes y dificultando mucho el entrenamiento. Con la reducción de la tasa de aprendizaje se puede, aunque no siempre, esquivar este problema.

Softmax:

A la hora de la elección de la función de las neuronas de salida, es importante saber que nos encontramos ante un problema de clasificación, y por lo tanto, es interesante el uso de funciones Softmax.

Tamaño de batch:

En primer lugar habría que definir lo que es un batch. Como se ha mostrado, los dos dominios con los que se va a trabajar son notablemente grandes y sería poco recomendable pasar como datos de entrada en cada ciclo de entrenamiento el total de las imágenes de cada dominio. El batch es un subconjunto del total de datos el cual se usa como entrada, en lugar de utilizar el

dominio entero. Por lo tanto, si por ejemplo estamos trabajando con CIFAR10, el cual cuenta con un total de 50000 imágenes para entrenamiento, se podría usar un batch de 1000 imágenes, las cuales serían la entrada de la red en cada ciclo. De ser así, en 50 ciclos, la red habría analizado el total del dominio.

¿Qué efectos tiene el tamaño del batch al entrenamiento?

El impacto del batch sobre las redes neuronales es algo muy a tener en cuenta. Se ha podido comprobar que, aunque se aumente el tamaño del batch, la tasa de aciertos en test no aumenta con él, sino que puede hacer que la red no sea capaz de adaptarse correctamente y su tasa de acierto no llegue a los niveles esperados. Si a esto se le suma que, al pasarle más instancias de entrada a la red por ciclo de ejecución, esta tiene unos tiempos de entrenamiento mucho más altos, hace que los tamaños de batch altos no tengan mucho sentido.

Learning rate	Batch size	Max word accuracy (%)	Training epochs
0.1	1	96.49	21
0.1	10	96.13	41
0.1	100	95.39	43
0.1	1000	84.13 +	4747 +
0.01	1	96.49	27
0.01	10	96.49	27
0.01	100	95.76	46
0.01	1000	95.20	1612
0.01	20,000	23.25 +	4865 +
0.001	1	96.49	402
0.001	100	96.68	468
0.001	1000	96.13	405
0.001	20,000	90.77	1966
0.0001	1	96.68	4589
0.0001	100	96.49	5340
0.0001	1000	96.49	5520
0.0001	20,000	96.31	8343

Figura 23. Tamaño batch vs número de iteraciones [23]

Cabe destacar que a medida que se aumenta el tamaño del batch, es necesario aumenta la tasa de aprendizaje. Si cada vez que se introduce como entrada un batch de un tamaño considerable no se aumenta esta tasa, la red no es capaz de adaptar correctamente sus pesos. Al introducir numerosos datos de entrada, es necesario que la red se adapte con mayor rapidez en cada uno de los ciclos, y de ahí que se aumente este valor.

12.2. CNN

Por parte de la CNN, se va a utilizar un modelo convencional relacionado a este tipo de redes. Su arquitectura constará de 2 capas convolucionales, cada una de ellas asociada a una capa de max_pooling. Como bien se sabe, las capas convolucionales, a diferencia de un MLP que no consta de las mismas, son las encargadas de la extracción de características de la imagen. La primera de estas capas tiene una dimensionalidad de 5x5x32, siendo las dos primeras dimensiones las encargadas de generar un kernel de 25 pixeles cuadrados encargados de la tarea de convolución, mientras que el ultimo valor es el numero de características que se quiere extraer, es decir, 32. La segunda capa convolucional tendrá las mismas dimensiones de kernel, pero en esta ocasión realizará la extracción de 64 características. Las capas de max-pooling asociadas a cada una de las convoluciones anteriormente mencionadas serán de dimensiones 2x2 en ambos casos. Con el fin de reducir el coste computacional, el kernel encargado de realizar el max-pooling en el caso de trabajar con el dominio CIFAR10 tendrá un tamaño de 3x3 . Por último, como capa final oculta se usará una capa totalmente conectada, equivalente a un MLP, de 1000 neuronas.

Como ya se sabe, es fundamental adaptar la red al dominio con el que se quiere trabajar. En el caso de las CNN, aunque tengan una arquitectura más compleja que los perceptrones multicapa, sus capas de salida y de entrada para cada dominio estarán dispuestas de la misma manera que en el apartado anterior.

A la hora de la elección de la función de activación se ha optado por el uso de ReLU, tanto en redes convolucionales como en la capa completamente conectada. En un principio se pensó en realizar algún tipo de prueba con alguna de las funciones de tipo sigmoidal o tangente hiperbólica. Sin embargo, si se basan los análisis en las explicaciones dadas en el Apartado 13.1, no tiene mucho sentido el uso de este tipo de función, y menos trabajando con un dominio significativamente más grande como es el CIFAR10.

A diferencia del MLP, aquí la función de optimización no será de descenso de grandiente, sino una de las funciones otorgadas por la librería tensorflow denominada AdamOptimizer [24]. Adam optimizer también está basado en el descenso de gradiente, pero incorpora algunas diferencias que, para el ámbito del deep learning, funcionan de manera satisfactoria. Este proceso para la optimización combina un algoritmo adaptativo del gradiente, con el algoritmo RMSProp [25], el cual es muy utilizado cuando se trabaja en el ámbito del Deep Learning. Se puede apreciar en la siguiente figura, como en el análisis del MNIST con redes profundas, se obtienen mejores resultados que con otros tipos de algoritmos de optimización.

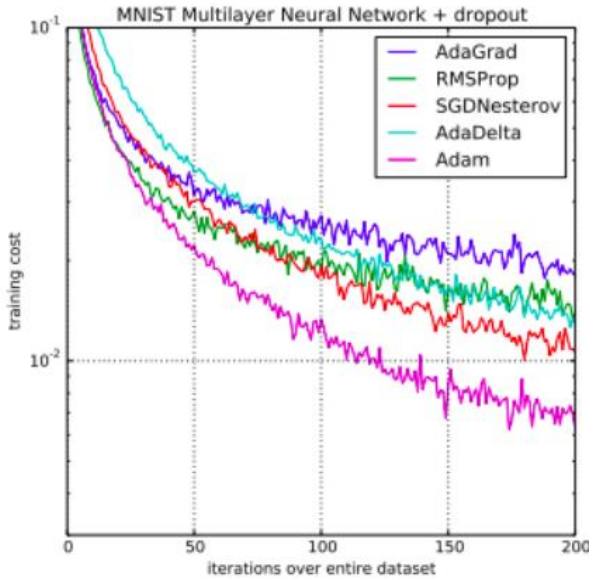

Figura 24. Comparativa entre algoritmos de optimización

Tamaño de batch:

Los tamaños de batch con este modelo es un tema en el que no se va a indagar tanto. El motivo es simple, como se ha podido apreciar en el anterior apartado en el que se realizaba el análisis del tamaño de batch con el MLP, el aumento considerable de su tamaño no influye en el porcentaje de acierto de la red a largo plazo, e incrementan notablemente los tiempos de ejecución. Por lo tanto, a pesar de que se realizará un pequeño análisis para comprobar el comportamiento al aumentar o disminuir su tamaño, se prefijará un volumen de 100 imágenes en cada minibatch.

Cabe destacar que con el uso del anterior mencionado algoritmo de optimización, AdamOptimizer, el cual cuenta con parte del funcionamiento de RMSProp, se obtienen resultados muy satisfactorios usando mini-batches. Se sabe que RMSProp funciona muy bien con batches de reducido tamaño debido a que no influye tanto el impacto del resto de pesos de la red, sino que teste algoritmo les da más importancia a los valores anteriores del propio peso.

13. EXPERIMENTACIÓN

Para este apartado se va a pasar a la realización de una serie de pruebas para poder llegar a comprender y profundizar en el aprendizaje de las redes neuronales explicadas con anterioridad.

Parámetros de las RNA que se modificarán para experimentar:

Nº de neuronas: El numero de neuronas que conforman cada una de las capas ocultas.

Nº de capas ocultas: Numero de capas con las que cuenta la red, sin tener en cuenta la capa de entrada y la de salida.

Learning rate: La tasa de aprendizaje en cada uno de los experimentos

Batch size: El tamaño de los bloques de datos de entrada en los ciclos de entrenamiento.

A la hora de realizar estas pruebas, inicialmente, se pasará a analizar distintos parámetros de la red para poder entender su comportamiento.

Precisión en entrenamiento: La precisión en entrenamiento indica el porcentaje de instancias de entrada bien clasificadas del mismo conjunto de entrenamiento.

Precisión en test: La precisión en test indica el porcentaje de instancias de entradas bien clasificadas con las que la red no ha entrenado. Es decir, mide la capacidad de generalización del modelo, debido a que se usan para calcularla patrones de entrada que la red nunca antes ha analizado. Por lo general este porcentaje de acierto debería ser menor que el de entrenamiento.

Cross-entrpy: Esta medida es muy interesante si queremos profundizar en el estudio de redes neuronales. Lo que nos mide el cross-entropy [26], en español la entropía cruzada, es la media de bits necesarios de modo que se pueda identificar un evento de entre un conjunto de posibilidades, si un esquema de codificación está basado en una distribución de probabilidad dada q, más que en la verdadera distribución p. Esta entropía calculada sobre dos distribuciones p y q sobre un mismo espacio de probabilidad se define como de la siguiente manera:

$$H(p,q) = E_p[-logq] = H(p) + D_{KL}(p \vee q)$$

Donde $H(p)$es la entropía de p y $D_{KL}(p \vee q)$ es la divergencia de Kullback-Leibler entre q y p, la cual es conocida como entropía relativa. Esta entropía describe la divergencia entre las probabilidades p y q, la cual puede venir dada por la distancia entre las dos distribuciones o o el numero de extra bits en muestras de código de P cuando se usa un código basado en Q, en lugar

de un código basado en Q. Generalmente es más correcta la segunda definición ya que generalmente se da que $D(p,q) \neq D(q,p)$. De este modo, gracias a cross entropy, se puede ver la diferencia entre la distribución esperada de la red y su distribución real.

Estos tres últimos valores, tanto la precisión en test, precisión en entrenamiento y cross-entropy serán representados en formato de grafica, debido a la facilidad de análisis que brindan las misma, y para una mejor explicación frente al lector. Hay que decir que se ha decidido, en algunas ocasiones, suavizar estas graficas. Esta es una de las ventajas que nos otorga tensorboard gracias a su interfaz de usuario. Esta opción nos permite trabajar con datos los cuales cuentan con numerosas oscilaciones y no se puede apreciar con facilidad la tendencia de los mismos.

Como última anotación antes de pasar al análisis de la experimentación es necesario apuntar que toda ella se realizó con un procesador Intel® Core™ i5-6200U 2..3GHz.

13.1. Carga y uso del dominio CIFAR-10

A la hora de realizar la carga del dominio del CIFAR-10 en el MLP surgieron una serie de complicaciones. A diferencia del resto de modelos, para este no había prácticamente ninguna información, ya que no es usual el uso de este tipo de redes con este dominio. Por lo tanto, se procedió a la descarga del dominio CIFAR-10 completo de la página oficial del mismo [21] en su versión para Python. Como ya se ha comentado anteriormente, este dataset cuenta con 60000 imágenes en total, de las cuales 50000 son para entrenamiento y 10000 para test. El dataset, en este caso en particular, se organiza en 5 batches de entrenamiento, cada uno de ellos con 10000 imágenes aleatorizadas, y otro batch para el testeo, con las 10000 imágenes restantes.

Un paso importante a tener en cuenta era el formato de los datos del dominio, para poder cargarlos de manera correcta, y que cada etiqueta se asignase correctamente a su imagen, sino el entrenamiento seria un proceso totalmente aleatorio. Según la información proporcionada por la página oficial [21], el dataset tiene una estructura de diccionario, en la que contamos con dos claves, “data” y “labels”. La parte de data, en cada uno de los batches, es un numpy array de 10000x3072, siendo cada una de estas filas del array cada una de las imágenes a color de 32x32 pixeles. Los primeros 1024 dígitos corresponden al color rojo, los 1024 siguientes al verde y los 1024 últimos al azul. Por otro lado en “labels” se halla una lista de 10000 posiciones, cada una de ellas con un valor entero de 0 a 9. Estos valores representan la clase de cada una de las imágenes anteriormente explicadas.

Como el MLP cuenta con una función de softmax al final para el cálculo de la probabilidad de pertenencia de una imagen a una clase, fue necesario hacer un pequeño cambio en el dominio, debido a que este no devuelve una lista de 10 posiciones, cada una de ellas con una probabilidad de pertenencia a cada clase, sino que devuelve directamente un numero entero. Lo que se hizo fue simplemente crear una función que transformaba el número entero en una lista de 10 posiciones con una probabilidad de pertenencia a esa clase del 100%. La transformación se vería de la siguiente manera: [6] = [0,0,0,0,0,0,1,0,0,0]

También fueron necesarias funciones las cuales eran encargadas de la actualización del batch que recibía la red neuronal.

En cuanto a la carga del dominio MNIST no se encontró ninguna dificultad.

13.2. Experimentación con MLP

En este apartado de experimentación cabe destacar que no se van a mostrar todas las pruebas que se han realizado debido a que son demasiadas. Aquí se mostrarán las más relevantes para el estudio que se llevará a cabo. Como ya se ha comentado, la experimentación inicial se realizará con la arquitectura explicada en el Apartado 12.1, haciendo uso del Teorema de Aproximación Universal. Se tendrá en cuenta el método Thumb y se usará una sola capa con 600 neuronas sobre el dominio del MNIST. Para el inicio de la experimentación con CIFAR10 la arquitectura constará de 3 capas ocultas cada una con 800 neuronas. Ambas arquitecturas serán fruto de los análisis realizados en el Apartado 15.2 y en base a ella se realizará el resto de análisis sobre los demás parámetros.

13.2.1. Desviación estándar en inicialización aleatoria de pesos.

¿En qué consiste este parámetro? En un inicio no se tuvo en cuenta este valor, debido a que se pensaba que este que no tendría ninguna influencia sustancial a la hora del entrenamiento de la red. Este parámetro impone una desviación estándar en la inicialización aleatoria de los pesos. Inicialmente en una red neuronal, el valor de los pesos no puede ser 0, debido a lo explicado en el Apartado 8.5. Como ya se sabe, un peso igual a 0 provoca que esa neurona quede inservible, debido a que al aplicar la regla delta su derivada siempre será 0. Por lo tanto, los pesos de una red neuronal se inicializan a valores aleatorios dentro de un rango predefinido. Al MLP que trabaja con MNIST se le impuso una desviación estandar de 0’03. Sin embargo, si se usa esta misma desviación con el CIFAR, la red no es capaz de aprender, a menos que se aumente de manera sustancial el número de iteraciones. Esto los será analizado posteriormente, sin embargo, probablemente esto sea debido a que si los valores a los que se inician pesos son más altos, la red tiene más probabilidades de no converger debido a que estos valores tienen inicialmente valores más dispares. Reduciendo el rango de posibles valores a los que inicializar los pesos a valores muy bajos, sin que estos lleguen ser a 0, se puede conseguir que la red comience a aumentar su tasa de acierto de manera sustancial. Para que la red sea capaz de realizar alguna modificación en los pesos sin que se estanque desde un inicio, es necesario reducir la tasa de aprendizaje a un valor de 0’00005, mientras que con la desviación estándar de 0’003 podemos aumentar la tasa incluso 100 veces más, y gracias a esto la red consigue mucho mejores resultados en un tiempo mucho menor.

13.2.2.Numero de capas ocultas y neuronas

Como base para la experimentación se realizará un análisis de la arquitectura a nivel de neuronas y capas ocultas. De este modo se elegirá la configuración más conveniente para cada modelo. Se ha mencionado en la introducción a la experimentación con el MLP, que en base al estudio previo que se realizó -Apartado 12.2-, se decidió seguir la idea del método Thumb, el cual nos da una idea aproximada de partida en cuanto al número de neuronas ocultas se refiere.

Hay que tener en cuenta que las imágenes del CIFAR son en torno a unas cuatro veces superiores que las del MNIST. De este modo se puede deducir que el espacio de búsqueda es mucho mayor que en el caso anterior, y el avance por dicho espacio ha de ser más minucioso, por lo tanto, esa tasa de aprendizaje habría que reducirla para que así los saltos de búsquedas sean menores.

A continuación se pasará a comprobar cómo se comportan con 3 capas ocultas –Figura 25-, cada una de ellas con 300 neuronas. Siguiendo el estudio teórico, el aumento del número de capas ocultas debería proporcionar a la red una mayor dimensionalidad a la hora de encontrar la función que mejor se ajuste. Así se podrá observar cómo actúan estas redes en comparación con las que solo cuentan con una capa oculta –Figura 26-.

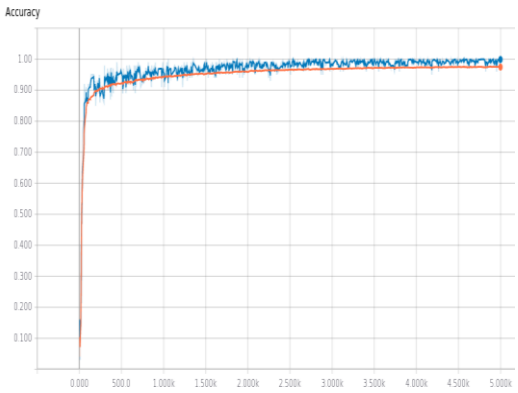

Figura 26. Una capa oculta

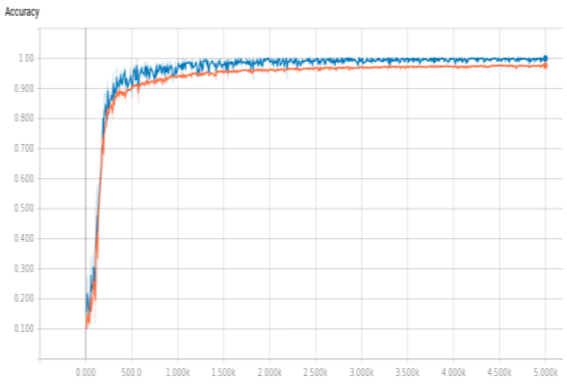

Figura 25. Tres capas ocultas

Se puede apreciar ligeramente al inicio de cada una de las graficas que, para este dominio en concreto, el aumento del número de capas ocultas provoca que la convergencia del modelo se produzca ligeramente más tarde. Si se aumenta el número de iteraciones de entrenamiento, ambos modelos acaban convergiendo al mismo acierto en test. Por lo tanto, para el dominio MNIST, la mejor elección parece ser la de una sola capa oculta.

A la hora de trabajar con CIFAR10 se necesita un mayor número de pesos para que la red sea capaz de extraer patrones debido a la diferencia de tamaño de dominio con respecto al MNIST,

lo que conlleva un número de neuronas más elevado. El uso del mismo número de neuronas que en MNIST lleva a unos resultados poco prometedores, produciendo estancamientos prematuros y porcentajes de acierto muy reducidos. Siguiendo el análisis realizado en el Método "Thumb", se podría empezar con un total de unas 2000 neuronas en una sola capa. Ha sido muy interesante constatar el impacto que tiene el tamaño del batch y la tasa de aprendizaje en la elección del número de neuronas. Se han realizado pruebas con 700 neuronas en una sola capa –Figura 27- y los resultados han sido que, al utilizar esta arquitectura, era necesario reducir drásticamente la tasa de aprendizaje para que el modelo no se estancase al inicio del entrenamiento, utilizando un 0'0001 de learning rate. Sin embargo, con 3 capas de 800 neuronas –Figura 28- y manteniendo un mini-batch de 100 imágenes, se podía aumentar el learning rate hasta un valor de 0'01, es decir, 100 veces superior. No es necesario un número de ciclos muy elevado para comprobar que modelo funciona mejor, por lo tanto se han realizado 2000 iteraciones en cada experimentación para ver cuál de estas redes es capaz de converger antes.

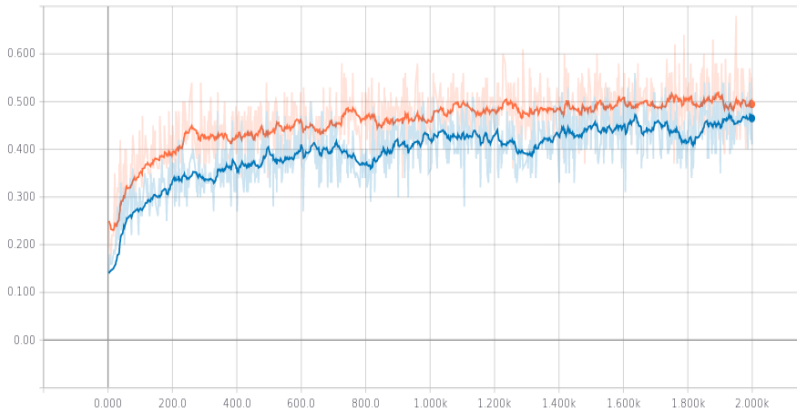

Figura 27. Una capa oculta

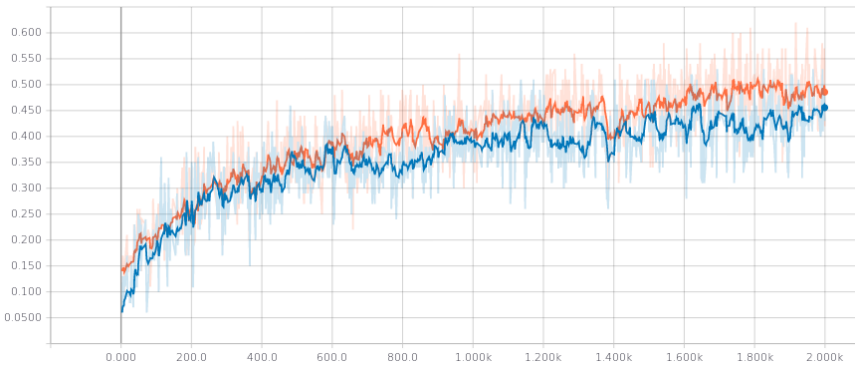

Figura 28. Tres capas ocultas

A simple vista, ambas graficas se asemejan bastante, aunque se puede apreciar que la convergencia a valores que rondan el 50% de acierto se producen antes cuando se tiene una sola capa en lugar de tener 3. Sin embargo, si se expone a la red a un entrenamiento con un aumento considerable de los ciclos de ejecución, 12.000, la red con una sola capa no era capaz de aumentar tanto la tasa de acierto como la red con 3 capas. A medida que se aumenta el número de ciclos para este dominio, las diferencia entre el acierto en test y en entrenamiento son más acusadas. Pero había una ligera diferencia entre ambas y es que en la red con más capas se alcanzaban valores algo más altos para el porcentaje de acierto en test. Esto puede ser debido precisamente a la dimensionalidad que otorga el aumento en cuanto al número de capas ocultas. Teniendo en cuenta los resultados obtenidos, el resto de experimentaciones mantendrán la misma arquitectura a nivel de neuronas y capas ocultas. El MLP con MNIST contará con 600 neuronas en una sola capa, mientras que con CIFAR10 tendrá 800 neuronas repartidas en 3, lo que hace un total de 2400 neuronas. Si se presta atención a la proporción

nº entradas/nº neuronas ambos modelos cuentan con un ratio de $\frac{Nº\ entradas}{Nº\ neuronas} \approx 1'3$

13.2.3. Tasa de aprendizaje

De las primeras experimentaciones sobre el dominio MNIST se podía concluir que el aprendizaje comenzaba a estancarse sobre los 600-800 ciclos, teniendo en cuenta que el mini-batch que se pasaba en cada ciclo de ejecución contenía 100 imágenes, y que la tasa de aprendizaje era de 0'1. La tasa de aprendizaje está ligada al tamaño del mini-batch. Si se aumentan los ciclos hasta los 2000, se puede comprobar cómo la red neuronal se estanca completamente. Eso quiere decir que se ha llegado a un mínimo local en la búsqueda de la minimización del error. Como se ha explicado anteriormente en el apartado de Regla Delta Generalizada –Apartado 8.5-, la tasa de aprendizaje va asociada con el "salto" que se produce en el espacio de búsqueda. Cuanto mayor sean esos "saltos" más probabilidades hay de saltar mínimos locales que no son solución óptima, sin embargo se pueden llegar también a resultados erróneos.

Con la configuración anterior se obtiene entorno a un 97% de precisión, la cual se podría mejorar aumentando esa tasa de aprendizaje, si lo hacemos, realizando un aumento hasta un 0'2 en dicha tasa, podemos comprobar lo siguiente –Figura 29-:

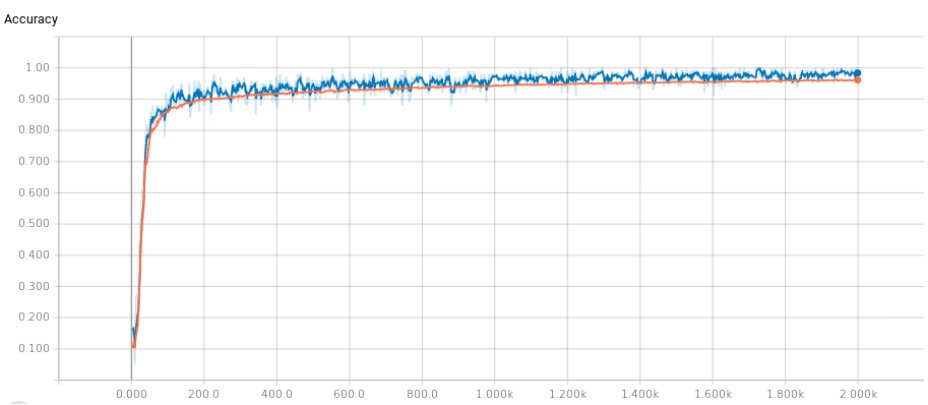

Figura 29. Aumento de tasa de aprendizaje a 0,2

Cuando se llega al ciclo 100 ya se alcanza un 90% de precisión. Se puede comprobar por lo tanto que la convergencia hacia valores altos de precisión es más prematura. Cuanto más se aumente esta tasa, también aumenta la variabilidad de la precisión, es decir, aparecen más picos en la función a medida que esta aumenta. Lo correcto sería escoger una tasa de aprendizaje que no fuera excesivamente alta, la cual vaya acorde con el número de ciclos escogidos para el entrenamiento.

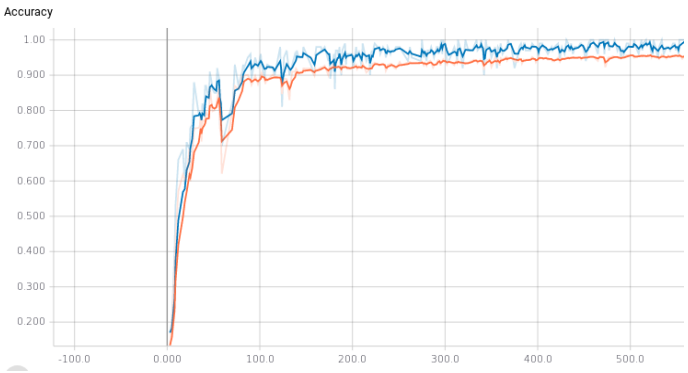

Figura 30. Aumento de tasa de aprendizaje a 0,4

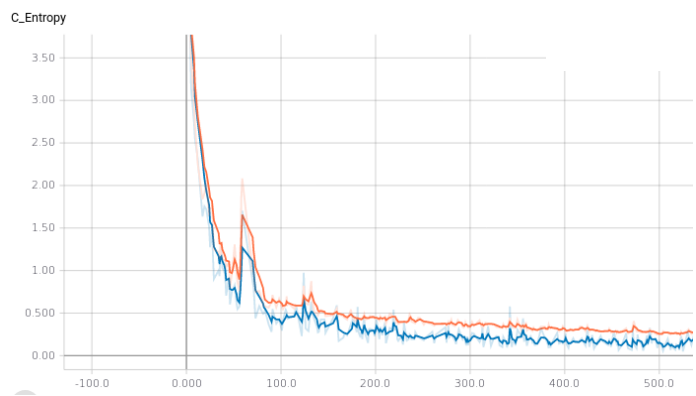

Figura 31. Cross-entropy con tasa de aprendizaje de 0,4

Se observa que los picos que generan las funciones, tanto en la tasa de acierto de test y de entrenamiento –Figura 30-, como en el cross-entropy –Figura 31- son mucho más acentuados. En el caso de estos dos ejemplos se utilizaba una tasa de aprendizaje de 0'4. Si se continúa aumentando ya no beneficia de ninguna manera al aprendizaje de la red, sino que más bien que estanca el mismo de forma muy prematura, dejando su porcentaje de acierto en torno a un 10% y de ahí no es capaz de salir.

Cuando se realizaron los experimentos con el MLP sobre el CIFAR10, se observo que las tasas de aprendizaje no podían ser tan elevadas. El motivo es básicamente los saltos que se realizan en el espacio de búsqueda. Un aumento en la tasa de aprendizaje lo que aumenta son dichos saltos, y hacen que la búsqueda sea más aleatoria. Por lo tanto, en un dominio como el CIFAR10, es necesaria una búsqueda más exhaustiva. De este modo, el uso de tasas de aprendizaje cercanas a 0'1 no llegan a la convergencia de aciertos en test.

Un concepto a tener en cuenta es la reducción paulatina del learning rate en el entrenamiento de las CNN con CIFAR10. Asumiendo que este comportamiento funciona razonablemente bien, mantener una tasa de aprendizaje más alta con un MLP sobre CIFAR10 no lleva a ningún resultado aceptable. Podemos ver en la Figura 32 como tras 11k iteraciones caen ambos aciertos. Las tasas con las que el modelo obtenía los mejores resultados antes de la caída de valores es 0'005.

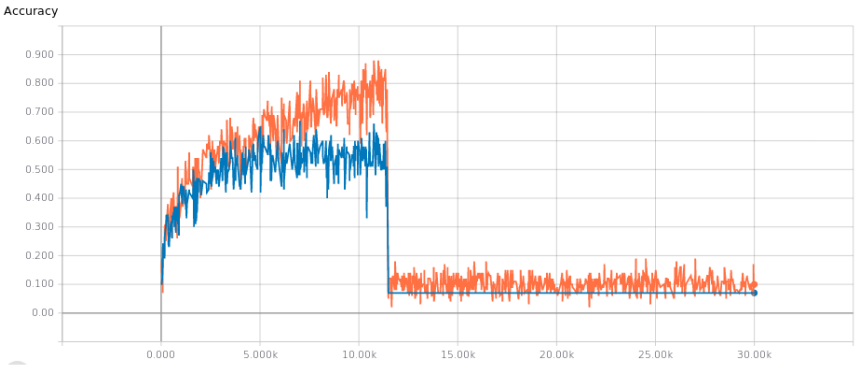

Figura 32. Caída del acierto

La tasa de aprendizaje va a depender de más valores que la arquitectura, y por lo tanto los resultados obtenidos no van a fijar dicha tasa para las experimentaciones que se muestran a continuación. Sin embargo este análisis proporciona una idea de cómo ajustar correctamente este valor para las siguientes pruebas.

13.2.4. Función de activación.

En cuanto a la función de activación se refiere, no tiene demasiado sentido el uso de la función sigmoidal, como ya se ha mencionado –Apartado 12.1-. La función más adecuada para estos dominios a clasificar con imágenes son las funciones de tipo ReLU. Sin embargo se va a comprobar de manera experimental que estas afirmaciones teóricas son ciertas. Si probamos con la anterior red con una sola capa oculta de 600 neuronas alternando entre la función sigmoidal y ReLU se podrá visualizar que con la función sigmoidal se necesitan llevar a cabo mas ciclos para poder converger en una tasa de acierto superior al 90%. Sin embargo, gracias a la función ReLU la convergencia se produce mucho antes. Como se puede apreciar en las dos gráficas que aparecen a continuación, se demuestra que la función de activación ReLU –Figura 34- es una alternativa más optima que una función sigmoidal –Figura 33-.

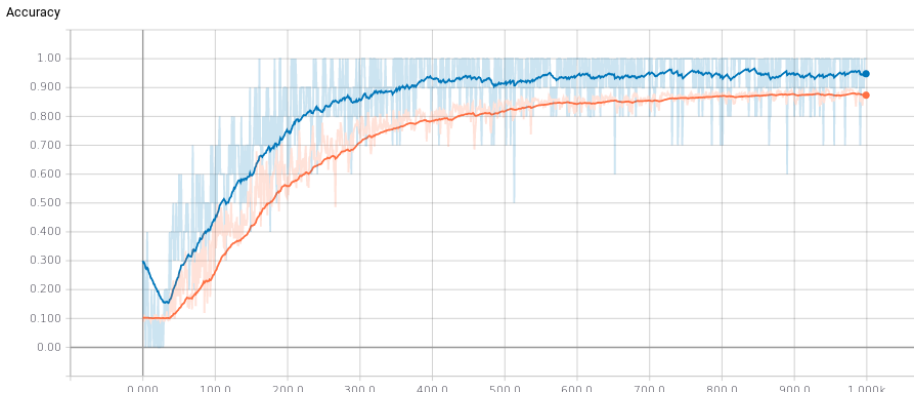

Figura 33. Función de activación sigmoidal

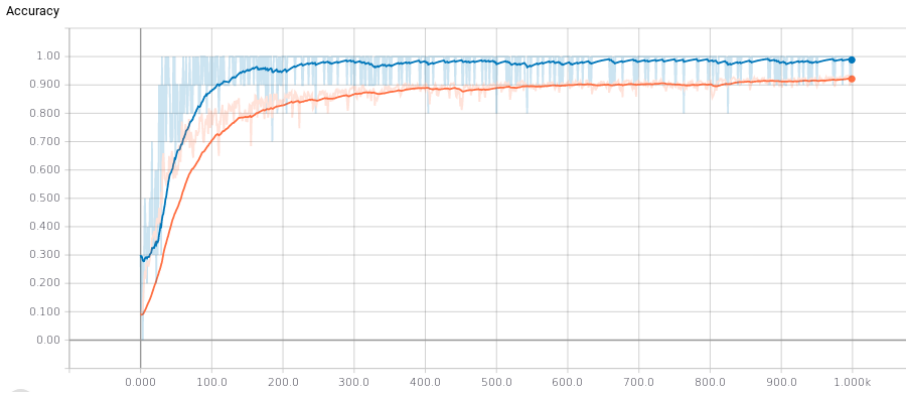

Figura 34. Función de activación ReLU

En el caso de trabajar con CIFAR10, la tasa de acierto usando una función sigmoidal se aleja mucho del MNIST, debido a que no se supera el 20% de acierto. Sin embargo, para este dominio hay una alternativa a la función de activación ReLU que resulta muy interesante, la Leaky_ReLU.

Como se ha definido en el Apartado 12.1, la función ReLU devuelve dos posibles valores, ó 0, o el propio valor del peso. Cuando se trabaja con un dominio tan grande como el CIFAR10, una de las causas por las que no se obtienen resultados satisfactorios es la "muerte neuronal". Esto ocurre cuando las neuronas devuelve como salida un valor de 0, y por lo tanto su derivada, en los siguiente ciclos de entrenamiento seguirá siendo 0, dejando esa neurona inservible. Teniendo en cuenta que la función ReLU puede llegar a lanzar este valor como salida de una neurona, es muy probable que a medida que se avanza en el entrenamiento, numerosas neuronas queden inservibles. Las ventajas que nos otorga la función Leaky_ReLU es que esta no puede devolver 0, sino un valor muy próximo sin este llegar a ser nulo. Por lo tanto es posible evitar este tipo de muertes neuronales.

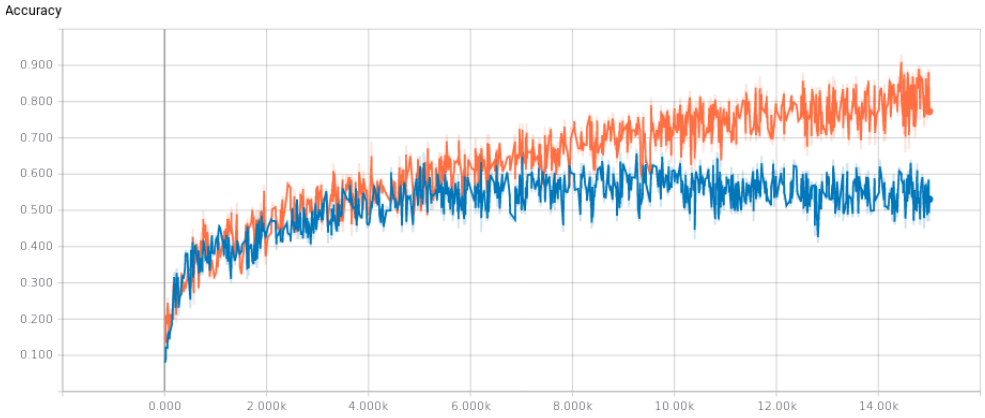

Figura 35. Función de activación ReLU

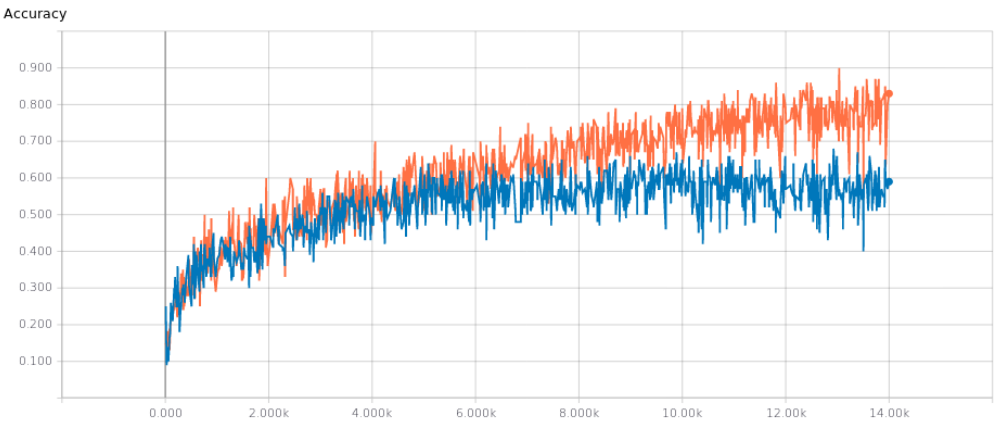

Figura 36. Función de activación Leaky-ReLU

Se puede apreciar que gracias a la aplicación de la función Leaky_ReLU -Figura 36-, el modelo llega a alcanzar valores cercanos al 67% de precisión en test, pero son picos esporádicos. Aun así, la convergencia hacia esos valores es más acentuada con Leaky_ReLU que con una función ReLU simple –Figura 35-, la cual se estabiliza entorno al 60%. No parece una mejora sustancial, pero la base teórica aplicada mejora levemente los resultados. Si se centra la atención en el número de iteraciones, el alcance de tasas superiores al 55% de acierto en test se producen antes con la función Leaky_ReLU.

13.2.5.Batch

Recordando lo mencionado en el apartado sobre el batch -Apartado 12.1- se sabe que a la hora de entrenar una RNA, a medida que se aumenta el tamaño del batch se ha debe aumentar también la tasa de aprendizaje. A continuación se van a mostrar una serie de experimentaciones para comprobar cómo se comporta el modelo a medida que se va aumentando el tamaño del batch y cómo, a su vez, se ha de incrementar también la tasa de aprendizaje. En ambos casos, se utilizarán las iteraciones de entrenamiento necesarias hasta que se comience a ver una convergencia, sin embargo, siguiendo los resultado de las experimentaciones anteriores, se mantendrán las funciones de activación que mejores resultados han proporcionado, siendo la función ReLU en MNIST y Leaky_ReLU con CIFAR10.

1 imagen:

Antes de examinar los resultados, se probó a entrenar la red con diferentes tasas de aprendizaje para este mini-batch en concreto, cuando se utilizaban tasas iguales o superiores al 0.1 el modelo se estancaba desde el inicio, por lo que se utilizó una tasa del 0.05.

MNIST:

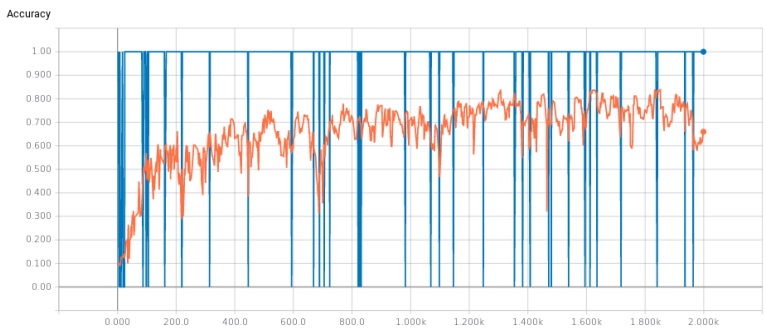

Figura 37. Mini-batch de una imagen

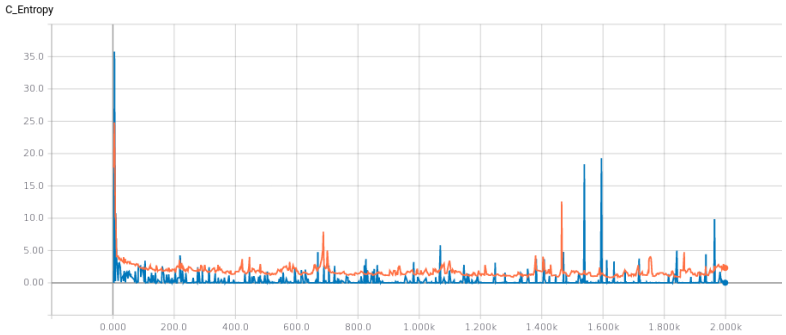

Figura 38. Cross-entropy mini-batch de una imagen

La tasa de aciertos en entrenamiento es muy alta ya desde el inicio pero con una gran cantidad de picos hacia abajo que llegan al 0% de tasa de aprendizaje. Esto puede ser debido a que en cada iteración únicamente se le está pasando una sola imagen, y el modelo no es capaz de generalizar y ajustar los pesos correctamente con tan pocos datos de una iteración a otra, ya que se observa que las bajadas de precisión no se mantienen, si no que son picos esporádicos. Otra de las causas relacionadas puede ser la tasa de aprendizaje, que aunque en un principio parezca pequeña, al estar tratando una sola imagen por ciclo puede ser algo elevada. Sin embargo, en los aciertos en test parece que la gráfica es un poco más regular, aunque también se destacan ciertas irregularidades. Si suavizamos la función –Figura 39- se podrá ver con más claridad la tendencia que llevan ambas tasas:

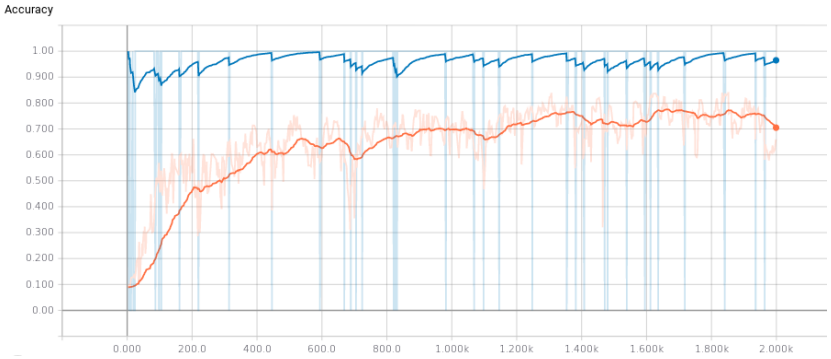

Figura 39. Suavización de la figura 37

Aunque el acierto en entrenamiento es bastante alto, el test no consigue los resultados esperados, sin llegar a alcanzar en las 2000 iteraciones una tasa superior al 80%. La gran diferencia entre entrenamiento y test llama enormemente la atención.

En cuanto al entrenamiento de la red sobre CIFAR10 no se han obtenido datos concluyentes debido a que el modelo no convergía a ningún valor específico, sino que simplemente oscilaba sin llegar a un porcentaje de acierto concreto.

10 imágenes:

MNIST

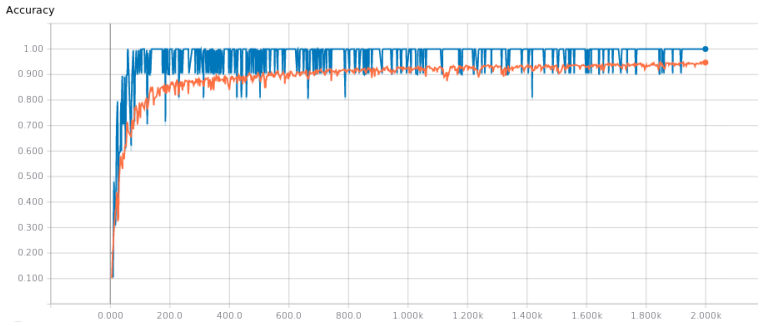

Figura 40. Mini-batch de 10 imágenes

Se puede apreciar en esta ocasión –Figura 40- que los picos de entrenamiento ya no son tan acusados, debido a lo que se ha comentado anteriormente. Al ser en este caso el mini-batch 10 veces mayor, la red, de una iteración a otra, es capaz de extraer más información. Con este aumento del mini-batch se consiguen porcentajes de acierto mucho superior que con tan solo 9 imágenes más por cada ciclo.

CIFAR10:

En este caso, para el correcto análisis de los valores de precisión fue muy necesaria la suavización de la gráfica –Figura 41-, a modo de poder observar la tendencia de cada una de las precisiones. Los resultados no son nada prometedores, aunque difieren mucho del uso de una sola imagen por mini-batch sobre este dataset.

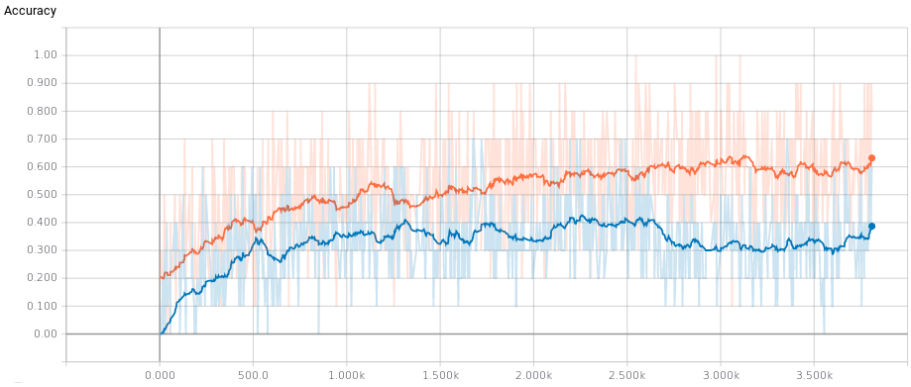

Figura 41. Gráfica suavizada mini-batch de 10 imágenes

100 imágenes:

Con este mini-batch de 100 imágenes, se produce una convergencia hacia valores que rondan el 97% de precisión en test antes de los 1500 ciclos de ejecución para MNIST. Teniendo en cuenta que el total de datos para entrenamiento son 60.000 imágenes, a los 1500 ciclos se habían pasado como entrada al modelo 1.500x100 = 150.000, es decir tan solo se ha pasado el dataset de entrenamiento completo en torno a 2'5 veces, alcanzando un porcentaje de acierto superior al 97% -Figura 42-. Esto da una idea de las ventajas a niveles de tiempo de ejecución que otorga el uso de mini-batches.

MNIST:

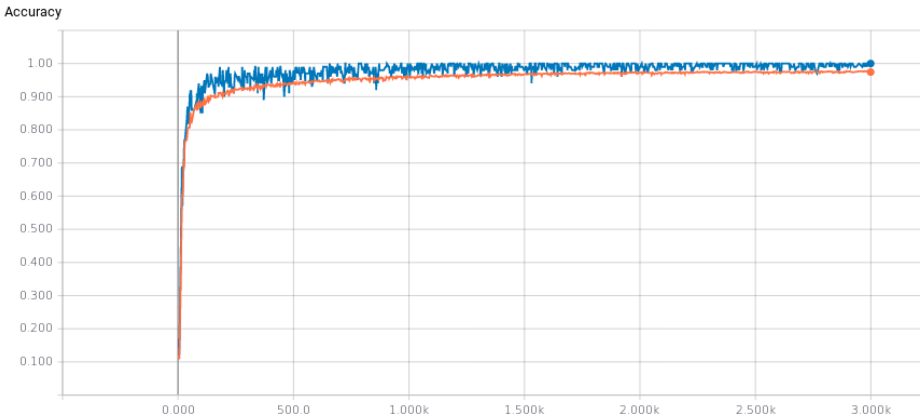

Figura 42. Mini-batch de 100 imágenes

La misma idea es aplicable al conjunto CIFAR10, aunque en este caso los resultados son mucho menos prometedores. Sin embargo, los valores más altos conseguidos en este caso por otros experimentos ajenos a este trabajo consiguen tasas de acierto cercanas al 57%, y esa cifra se alcanza a partir de los 2000 ciclos. Se ha procedido a la suavización de la gráfica –Figura 43- para una mejor visualización de su tendencia.

CIFAR10:

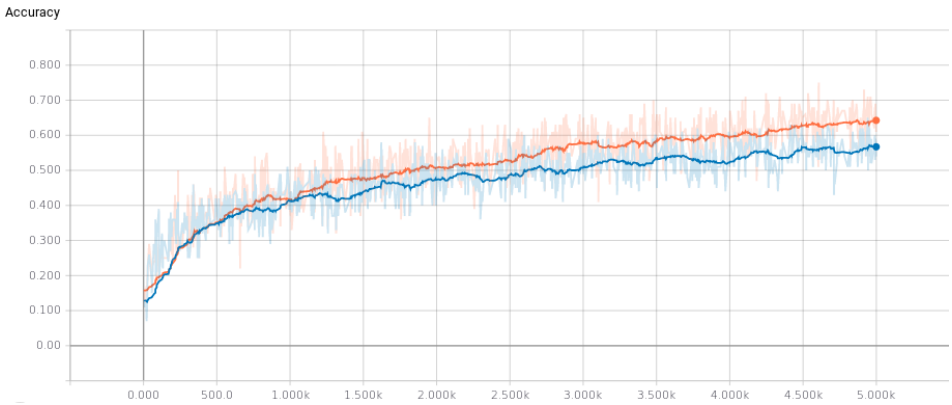

Figura 43. Mini-batch de 100 imágenes

En cuanto al aumento de más de 100 imágenes por cada mini-batch se refiere, se ha decidido no incidir demasiado, debido a que el aumento del tamaño de estos mini-batch no llevaba a ninguna mejora en cuanto a resultados, y aumentaba drásticamente los tiempos de ejecución. La convergencia a tasas de acierto más altas se produce mucho antes utilizando mini-batches más pequeños, incluso estos llegan a mejorar los resultados.

Parece que esta configuración de mini-batch es la más óptima en cuanto a precisión vs tiempo de ejecución. Si aumentamos de forma notoria el número de ciclos de entrenamiento podemos llegar a alcanzar tasas de acierto en test de entorno al 67%.

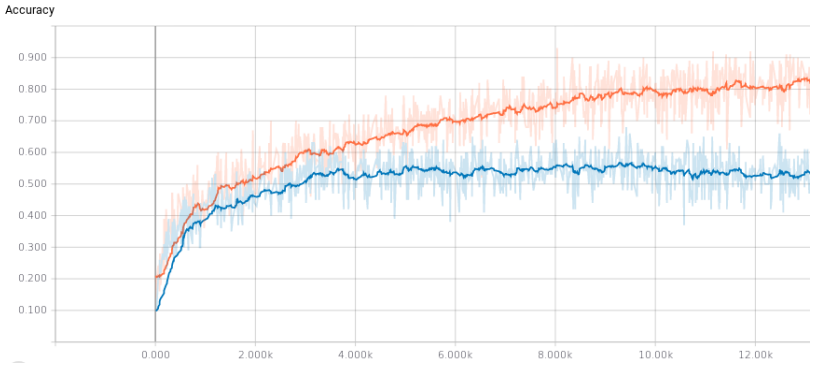

Figura 44. Mejores resultados de MLP sobre CIFAR10

Se puede apreciar como a entorno a los 6000 ciclos de ejecución, la precisión de test ya se estanca, la red no es capaz de ir más allá, mientras que el entrenamiento puede superar el 90% con el avance de las iteraciones. Aunque la gráfica –Figura 44- este suavizada, en numerosas ocasiones del entrenamiento, la correcta clasificación de patrones ha superado el 66% en números picos a lo largo del entrenamiento, siendo este el mejor resultado conseguido en este análisis para este modelo trabajando sobre CIFAR10.

La siguiente tabla que se muestra en el apartado 15.6 contiene un resumen de los mejores resultados obtenidos en las experimentaciones sobre ambos datasets. Más adelante –Apartado 18- se llevará a cabo un análisis más profundo de dichos resultados. Como nota aclaratoria, los siguientes aciertos que se muestran son los picos más altos obtenidos a los largo del entrenamiento, teniendo en cuenta que no sea un pico esporádico y la tasa tras el decaiga alarmantemente.

13.2.6. Resumen de la experimentación con MLP:

Modelo	Nº capas	Neuronas por capa	Función activación	Learning rate	Tamaño mini-batch	Función minimización error	Ciclos
MLP_MNIST	1	600	ReLU	0'2	100	Descenso de gradiente	3000
MLP_CIFAR10	3	800	Leaky_ReLU	0'005	100	Descenso de gradiente	14000

Modelo	Tasa acierto	Tiempo requerido
MLP_MNIST	0.987	-15 min
MLP_CIFAR10	0.672	- 3 horas

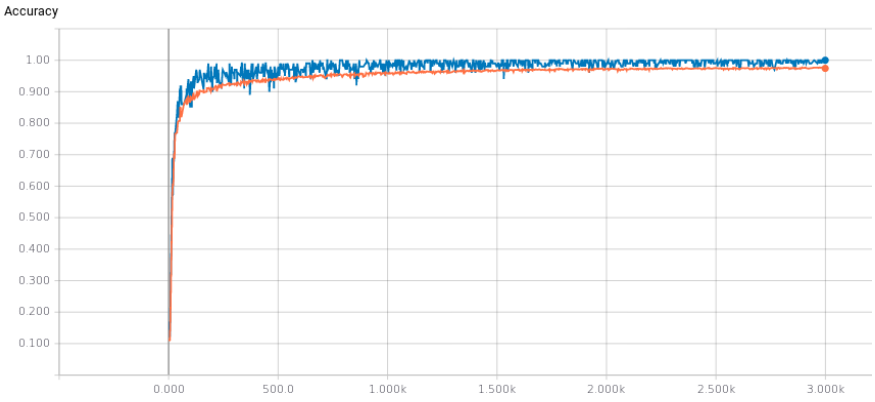

Figura 45: MLP_MNIST

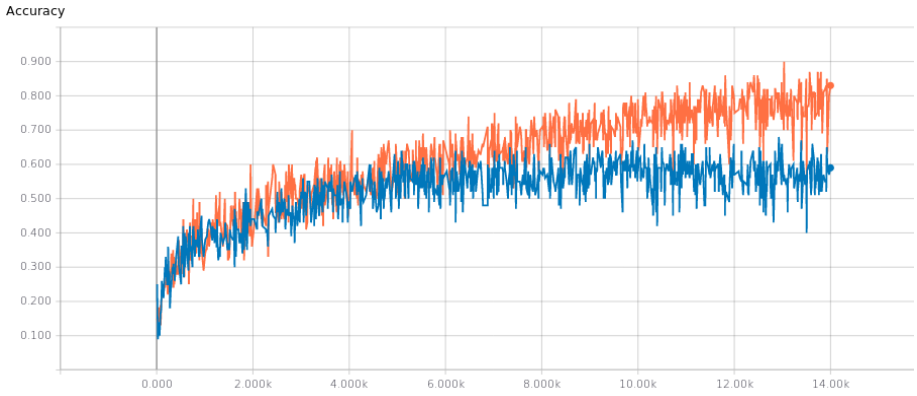

Figura 46: MLP_CIFAR10

13.3. Experimentación con CNN

A la hora de realizar la experimentación cabe destacar que no se han analizado los mismos parámetros que con el MLP. La experimentación centrada en el perceptrón multicapa ha sido mucho más exhaustiva de lo que será en el siguiente apartado. El motivo que ha llevado a esta decisión es principalmente la cuestión de que, a la hora de trabajar tanto con MNIST como con CIFAR10, se sabe que en ambas ocasiones se obtienen porcentajes de acierto muy satisfactorios con el uso de las CNN's. Sin embargo, como se ha podido comprobar, el problema del uso de un perceptrón multicapa sobre CIFAR10 no está resuelto, y uno de los objetivos fundamentales del análisis que se lleva a cabo en este trabajo es intentar conocer el porqué. No será necesaria la modificación de todos aquellos parámetros analizados a lo largo de la experimentación anterior debido a que, en base a todas las investigaciones que se pueden encontrar, es bien sabido que los resultados de estas redes es muy satisfactorio tanto en MNIST como en CIFAR10. El estudio por lo tanto en este apartado irá más dirigido a comentar, con un carácter más general, resultados obtenidos por este modelo neuronal, si bien el apartado anterior fue un intento por mejorar el comportamiento del MLP. Es decir, la experimentación que se realizará con este modelo no será tan extensa, si no que más bien se han realizado las experimentaciones necesarias para verificar que los resultados de este modelo son más precisos que los de un MLP.

En cuanto a los parámetros a analizar no ha sido necesario realizar experimentaciones con por ejemplo funciones del tipo sigmoidal, debido a que se podían obviar ya que lo que se pretender es ahondar en el comportamiento de estas redes cuando devuelven los mejores resultados y saber porque obtienen estos mismos.

13.3.1. CNN sobre MNIST:

Teniendo en cuenta que la experimentación con MLP ha dejado más que demostrada su eficiencia frente a MNIST, se puede presuponer que el uso de una arquitectura de red convolucional dará mejores resultados, aunque parece complicado pensar que se pueda superar una tasa de acierto superior al 98% en test.

La arquitectura de las CNN, la cual va dirigida a la extracción de características basada en el carácter convolucional de la misma, es capaz de superar los resultados de un MLP, aunque no de manera drástica debido a que este ya obtenida unos resultados muy satisfactorios. El gran problema surge cuando se enfrenta una red como la CNN, con una complejidad superior al MLP, a un dominio como MNIST el cual es bastante sencillo. Se podría decir que es un modelo demasiado sofisticado para que sea aplicado a este dataset. De todos modos se han realizado algunas pruebas para verificar su eficacia y comprender mejor su comportamiento.

13.3.2.Tasa de aprendizaje:

Aunque ya se haya especificado que el análisis de los parámetros estudiados en el MLP no ha sido necesario, la tasa de aprendizaje es uno de los obligados. Se quiso probar el porqué estas redes no soportan tasas de aprendizaje tan elevadas como el MLP. La cuestión, es que la cantidad de pesos con la que trabaja estas redes implica un descenso obligado en este valor. Es difícil pensar que con tasas altas, tal número de conexiones lleguen a estabilizarse para obtener resultados óptimos.

Los dos mejores resultados que se obtuvieron fueron con tasas de 0,0001 –Figura 47- y 0,001 –Figura 48- . A partir de ahí, el aumento de las mismas no llevaba a ninguna convergencia. Como era de esperar, la aplicación de un 0,001 propiciaba que la red convergiese antes a valores altos de acierto, sin embargo hacia que la variabilidad de dicho acierto fuese más elevada (se puede apreciar que los picos son más acusados) y que el resultado final no fuese tan cercano al deseado.

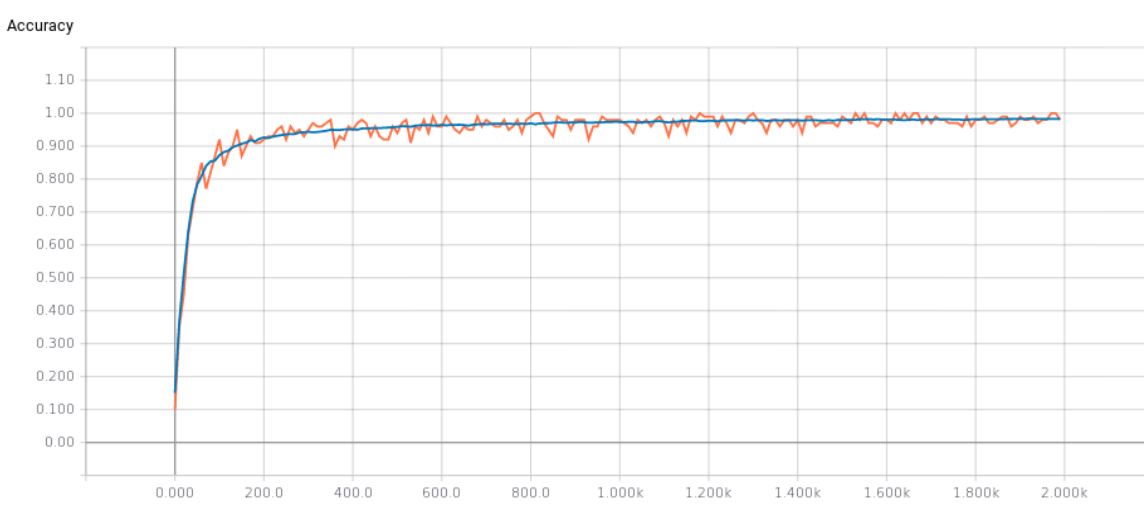

Figura 47. Tasa de aprendizaje de 0,0001

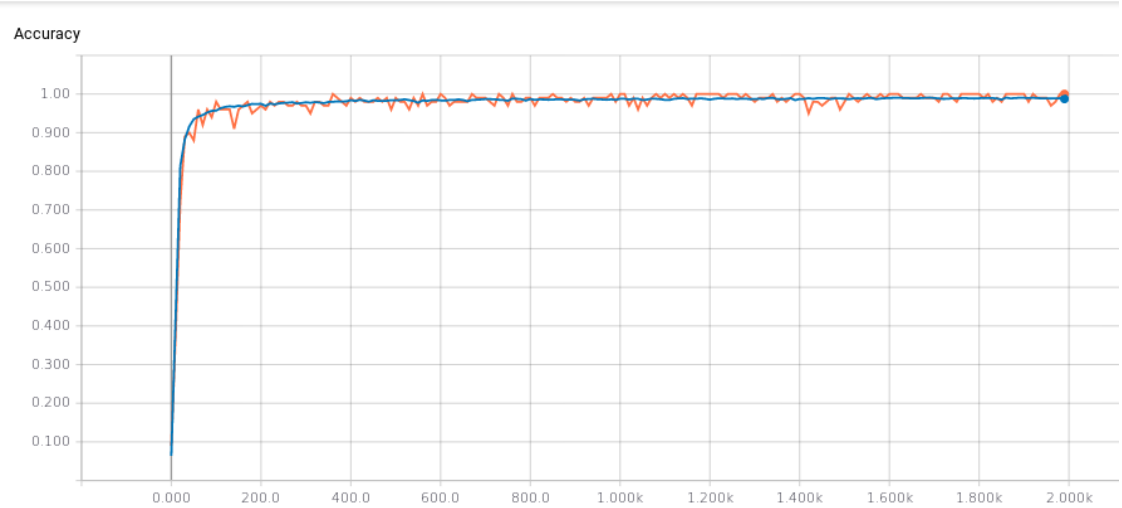

Figura 48. Tasa de aprendizaje de 0,001

Al inicio del entrenamiento se esperaba que la convergencia se produjese a distintas alturas, sin embargo se comprobó que el ligero aumento de la tasa de aprendizaje no producía ningún obstáculo con el avance de los ciclos. Lo esperado era que con una menor tasa de aprendizaje, aunque la convergencia fuese más costosa, se llegase a porcentajes de acierto más altos. En cambio se puede apreciar que ambos modelos obtenían los mismos resultados, sin embargo en la Figura 48 se observa que se obtienen antes tasas cercanas al 99%.

13.3.3. Función de minimización del error:

Se quiso analizar el impacto que tiene esta función en este tipo de modelos para comprobar si su utilización estaba justificada. Se pudo comprobar que la función AdamOptimizer era la mejor opción de entre las estudiadas. El uso del descenso de gradiente no aportaba ningún valor, debido a que cuando se realizaron las experimentaciones los resultados fueron nefastos.

No se ha barajado la opción de integrar dicha gráfica debido a que su tendencia era totalmente plana. La precisión en test no era capaz de aumentar a lo largo del entrenamiento.

13.3.4. Tamaño de batch:

No hay demasiado que decir acerca de este parámetro. En los análisis realizados al MLP se pudo observar como los mini-batches proporcionan mejoras frente a los batches de gran tamaño. Si se extrapola a las CNN es obvio que se siga la misma estrategia debido a que los entrenamientos con estas redes son mucho más costosos computacionalmente.

Los batches de gran tamaño tienen más sentido con dominios más grandes, en los que se necesita que los pesos de la red se adecuen con más cuidado, ya que se pudo observar en las Figuras 37 y 40 que una reducción considerable de estos mini-batches hacia que las salidas fuesen demasiado oscilantes.

Por lo tanto no interesa trabajar ni con tamaños de mini-batch excesivamente pequeños ni demasiado grandes. Para comprobar que con estos modelos también es beneficioso el uso de mini-batch de tamaños cercanos a 100, se realizó una prueba con un mini-batch con 1000 imágenes. Dichos resultados se pueden observar en la Figura 49.

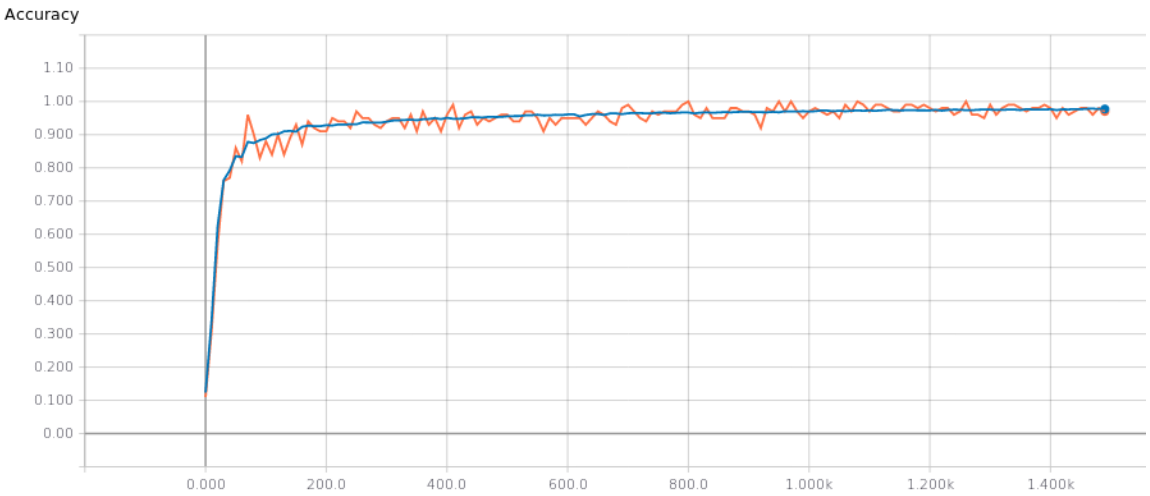

Figura 49. Mini-batch de 1000 imágenes

Si lo comparamos con la Figura 48, en la que se utilizó la misma tasa de aprendizaje (0,001), el uso de un mini-batch de mayor tamaño no implica mejoras. Se puede apreciar que aunque ambos modelos obtengan resultados muy similares, la convergencia hacia valores cercanos al 99% de precisión se produce mucho antes con mini-batches de tamaño 100.

13.3.5. Numero de neuronas en la capa totalmente conectada:

Aunque la idea inicial era realizar la experimentación con 1000 neuronas en la capa totalmente conectada, se decidió realizar un primer experimento para comprobar si se podían ahorrar recursos computacionales reduciendo este tamaño neuronal. Por lo tanto propuso un modelo en que se contaban con la mitad de neuronas en dicha capa (un total de 500). Los resultados fueron más que satisfactorios.

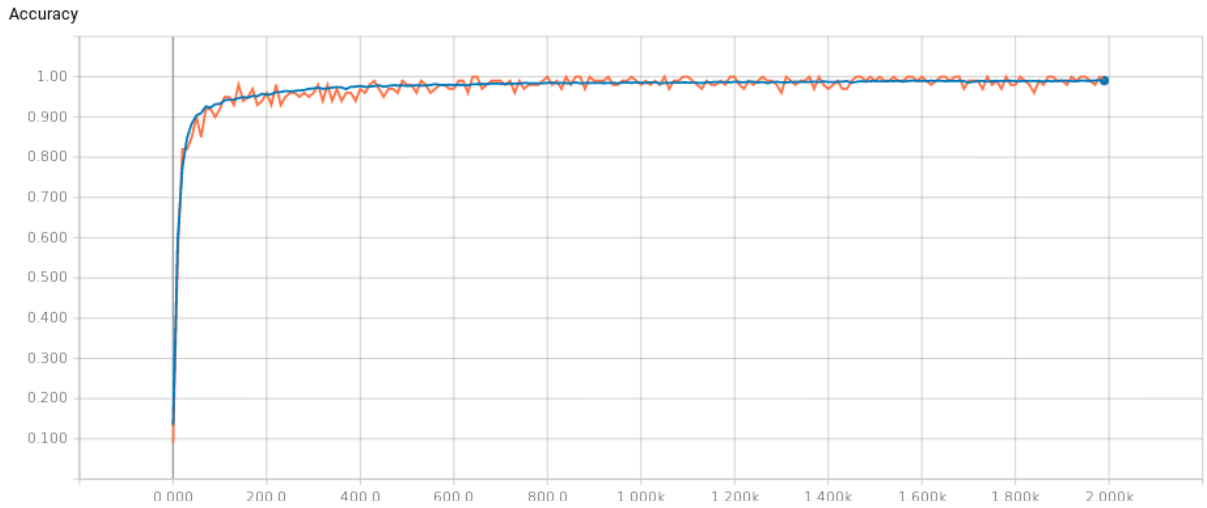

Figura 50. Capa totalmente conectada de 500 neuronas

Si lo comparamos con anteriores gráficas en las que sí que se usaba un tamaño de 1000 neuronas -Figura 48- los resultados son muy similares, sin embargo hay que meditar acerca del coste computacional. Con este modelo el entrenamiento fue ligeramente menos costoso debido al menor número de neuronas en la capa totalmente conectada, alcanzando a su vez una tasa de entorno al 99%. La convergencia hacia ese valor también se produce a la misma altura, por lo tanto podríamos decir que este modelo con tan solo 500 neuronas en esa capa es más eficiente que el anterior.

13.3.6. CNN sobre CIFAR10:

En este apartado se podrá comprobar el porqué del uso de este tipo de redes frente a dominios más complejos. Un dataset como MNIST no necesita de la extracción de características para poder ser clasificado de una manera optima. La diferencia entre número manuscritos es insignificante en comparación con imágenes a color de objetos y animales totalmente excluyentes entre ellos, como ya se ha comentado. Entre las diferentes clases, las imágenes no guardan ningún tipo de relación.

Los entrenamientos con estos dominios y redes son notablemente más costosos que en todos los casos anteriores. Como se ha indicado anteriormente, con un tamaño de mini-batch más reducido se consigue que el entrenamiento no sea tan pesado computacionalmente y se llegue antes a tasas de acierto más altas, por lo tanto se mantendrá a lo largo del análisis un tamaño de 100 imágenes.

En cuanto al número de neuronas en la capa totalmente conectada no se apreciaron mejoras al reducir su tamaño, por lo tanto tampoco se contempló mostrar la experimentación con dicho parámetro.

13.3.7. Función de minimización del error:

Como se ha analizado este parámetro en el anterior apartado, se ha decidido también realizar una pequeña prueba inicial, para fijar esta función. Se decidió hacer una comparativa entre AdamOptimizer y el descenso del gradiente. Los resultados no pudieron ser más sorprendentes. A diferencia del anterior caso, el descenso de gradiente funcionaba excepcionalmente bien, en contraposición del AdamOptimizer, que comenzaba con tasas de entropía muy altas. Gracias a cross_entropy se puede observar como el descenso de gradiente funciona significativamente mejor que AdamOptimizer, debido a que el acierto en test no es capaz de mostrar una comparativa real en el inicio de ambos entrenamientos.

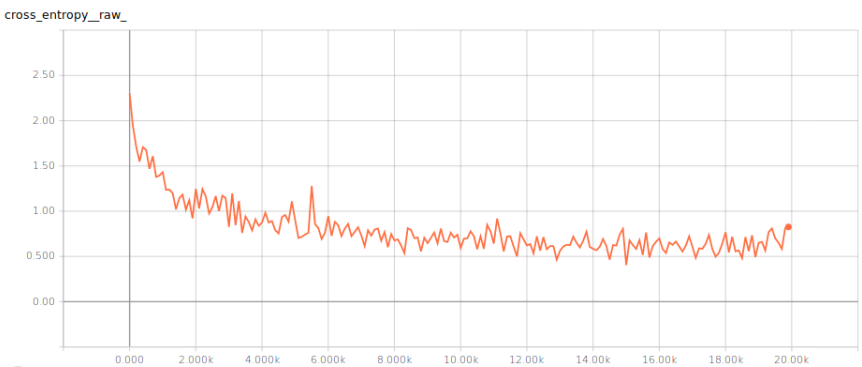

Figura 51. Función de minimización del error Gradient Descent

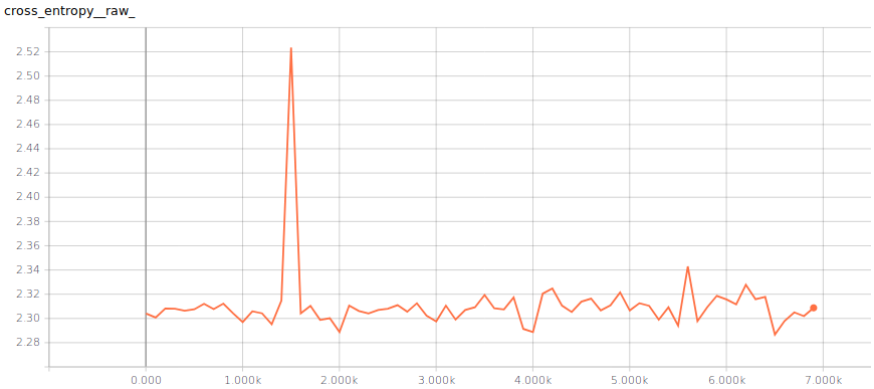

Figura 52. Función de minimización del error AdmOptimizer

En el caso de la Figura 52 no se necesitaron más de 7000 iteraciones para poder observar que el modelo el cual utilizaba AdamOptimizer funcionaba notablemente peor que con descenso de gradiente. Por lo tanto a partir de este punto se fijó como función de minimización del error esta ultima –Figura 51-.

13.3.8. Análisis general del modelo:

Se necesitaron más de 6 horas para alcanzar un 84’4% de instancias clasificadas de manera correcta. Con el uso de GPU y un entrenamiento de unas 5 horas se alcanzan tasas de acierto del 86%. Pero este resultado no es una cifra significativamente diferente a la anterior. Una CNN con esta tasa de acierto sobre este dominio es un resultado aceptable, sin embargo podría ser mucho más óptimo.

Aunque en el Apartado 13.2 se adelantaba la arquitectura que se iba a utilizar en esta experimentación, cabe destacar uno de los parámetros más importantes, la tasa de aprendizaje. En la anterior experimentación con el MLP se pudo comprobar cómo, con el aumento del número de neuronas y el amplio espacio de búsqueda del CIFAR10, es necesaria la reducción de la dicha tasa. Sin embargo, con las CNN, en numerosas ocasiones se introduce un concepto nuevo el cual se denomina decaimiento de la tasa de aprendizaje. Se muestra a continuación el uso inicial de un 0.1 de tasa de aprendizaje, con un acierto en test del 84.4% -Figura 53-.

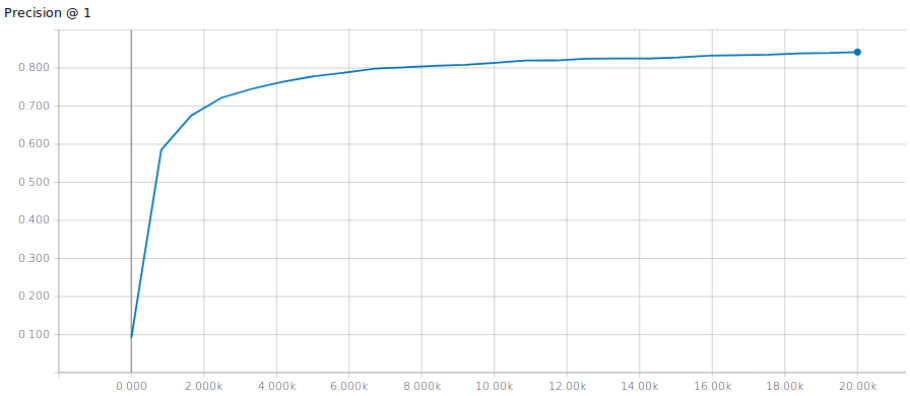

Figura 53. Tasa de aprendizaje inicial de 0,1 con Learning Rate Decay

¿En que se basa? El decaimiento de la tasa de aprendizaje está basado en la reducción paulatina en función número de ciclos de entrenamiento ejecutados. Por lo tanto, para las primeras iteraciones se comenzaba con una tasa del 0.1 y en función del número de iteraciones se irá reduciendo. ¿Qué beneficios puede reportar este comportamiento? La idea detrás de esto es que al comienzo del entrenamiento interesa que los saltos dentro del espacio de búsqueda sean grandes para poder converger antes a la zona donde se halle la solución óptima. A medida que el modelo se acerca a dicha solución, es más conveniente que esos saltos se reduzcan para aproximarse lo más posible sin realizar oscilaciones entorno a la solución sin llegar a aproximarse a ella.

Si con una CNN en un dominio como este se mantiene esa tasa, el modelo comenzará un entrenamiento normal, pero con el paso de los ciclos no será capaz de converger a ningún resultado, debido a que los saltos en el espacio de búsqueda serán demasiado grandes.

Se pasa a mostrar en la Figura 54 un entrenamiento sin ese decaimiento de la tasa de aprendizaje para ver si realmente es necesaria.

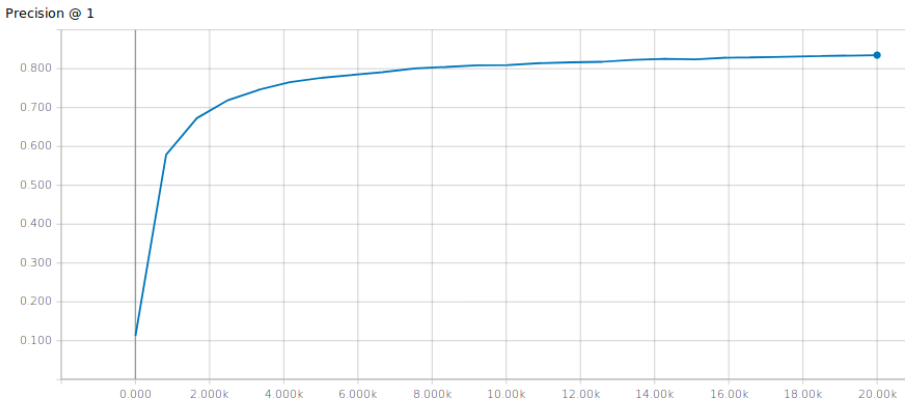

Figura 54. Tasa de aprendizaje constante de 0,1

A primera vista no se aprecia una diferencia muy acusada. Sin embargo el porcentaje de acierto en test es menor (en torno a un 82%) debido precisamente a que, si no se reduce la tasa de aprendizaje acorde con el avance del entrenamiento, al inicio el modelo no variará su comportamiento, pero según se avanza en el mismo el modelo no será capaz de acercarse con tanta precisión a la solución optima, debido a los saltos realizados en el espacio de búsqueda.

Debido a este decaimiento, no se vio la necesidad de la realización de más experimentos, ya que se utilizan en la misma experimentación varias tasas de aprendizaje.

Esta misma experimentación se realizó para comprobar su funcionamiento en un perceptrón multicapa. Sin embargo no se mostraron estos resultados debido a que no se pudo apreciar ninguna mejora sustancial. Esa arquitectura no necesita un decaimiento de ese tipo, debido a que la carga de pesos del modelo no es tan grande como en una CNN. El uso de este concepto sobre el dominio MNIST con las CNN tampoco tuvo efectos beneficiosos.

Como ya se ha mencionado, de estas experimentaciones se extrajeron otros datos que podían ser de valor a la hora de conocer cómo se comporta una CNN y ahondar en el porqué de su diferencia de resultados respecto al MLP.

Sparsity:

¿En que se basa el sparsity asociado a cada una de las capaz convolucionales? Es una medida que nos da una idea de las matrices que adquieren valor cero en cuanto a pesos neuronales se refiere. Cuanto más se avanza en la profundidad de la red, mayor numero de estos pesos se vuelve nulo. Este era el problema ocurrido en el MLP cuando trabajaba con CIFAR10, el cual se intentó solventar, en mayor o menor medida, con las funciones de activación del tipo Leaky_ReLU. Sin embargo con este modelo, el uso de este tipo de función de activación, aunque evitase este concepto de sparsity, no introducía grandes mejoras en los resultados obtenidos. Por lo tanto se mantuvo el uso de la función de activación del tipo ReLU en la experimentación.

En el Apartado 10 ya se habló de que, uno de los avances introducidos por estos modelos son las convoluciones. Estas capas encargadas de la extracción de características aportan una capacidad de generalización extra que la arquitectura de un MLP no soporta. Por lo tanto parecía interesante averiguar qué ocurría con estas capas.

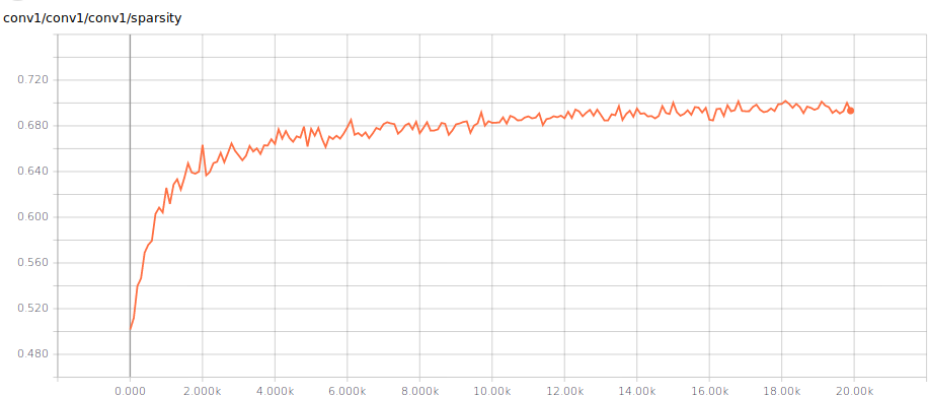

Figura 55. Sparsity en la primera capa convolucional

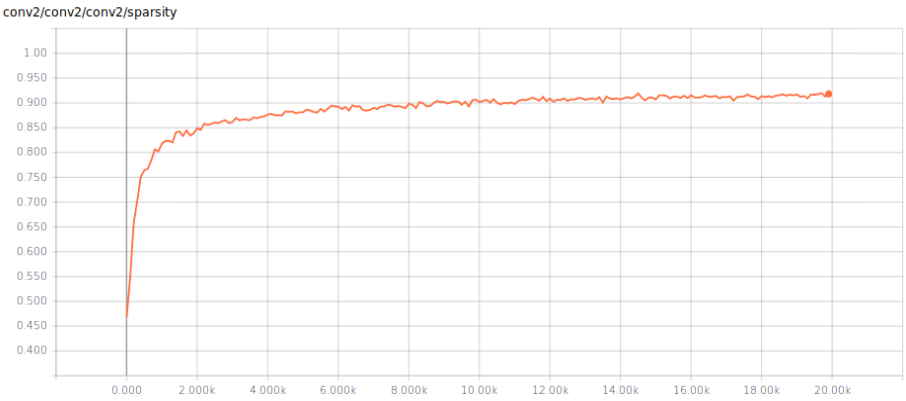

Figura 56. Sparsity en la segunda capa convolucional

Se puede observar en la Figura 55 como este valor no es tan acusado como en la siguiente capa convolucional. Tal y como se introdujo en el Apartado 9.2, la primera capa extrae un total de 32 características y en la segunda 64. Cuanto mayor es el número de conexiones entre neuronas, mayor es la probabilidad de que sus pesos sean 0 y dichas neuronas queden inutilizadas. Aun con estos grados de sparsity, las redes convolucionales siguen obteniendo mejores resultados que un MLP, lo cual es cuanto menos extraño.

Loss:

¿Qué es y cómo se interpreta este valor? Al igual que las tasas de acierto, el "loss" es un valor que mide como de bueno es un modelo clasificando las instancias de entradas. Sin embargo, a diferencia de la tasa de acierto, este valor no es un porcentaje, sino que representa la suma de los errores cometidos a la hora de la clasificación.

Por lo tanto, cuanto menores sean estos valores, mejor será el modelo entrenado. Aunque no guarda una relación directa con la tasa de acierto, están bastante ligadas. Teniendo en cuenta que este tipo de redes alcanzan una tasa de acierto en entrenamiento superior al 90%, se puede

apreciar en la Figura 57 que el error que nos representa "loss" confirma este porcentaje. Cabe destacar que no es el "loss" asociado a ninguna capa en específico, si no que está calculado con respecto a la totalidad del modelo, para tener así una visión más general.

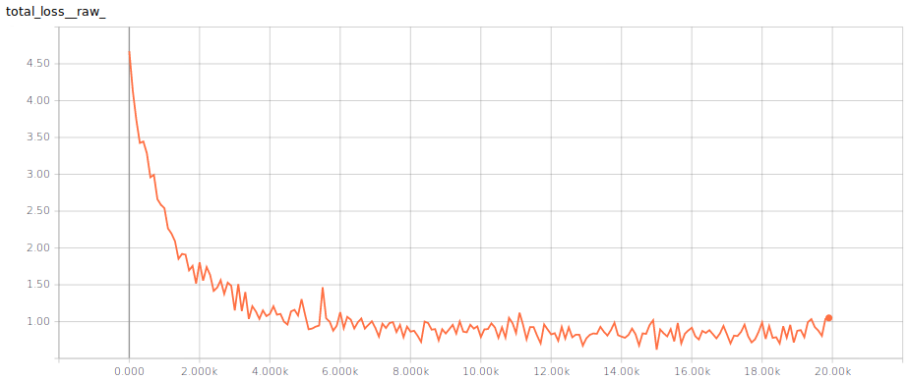

Figura 57. Loss de la CNN

Si el lector se fija con detenimiento en la gráfica, como suele ocurrir con este tipo de gráficas, al inicio se puede apreciar una gran caída en la suma de los errores en la clasificación de instancias, para que posteriormente se produzca un estancamiento hacia valores de error más bajos.

Esta gráfica fue de gran ayuda a la hora de comprender mejor cómo se comporta el modelo en base a una tasa de aprendizaje específica. Si se presta atención a la parte en la que se hablaba sobre el decaimiento de este valor, se afirmaba que si este concepto no era aplicado, el modelo se vería con muchas dificultades para aproximarse de manera más precisa a la solución optima.

Haciendo énfasis en el "loss" asociado al entrenamiento se puede ver una gran variabilidad a lo largo del mismo. Estos picos son normales al inicio del entrenamiento, sin embargo a medida que se avanza en el mismo sería necesario la reducción de la tasa de aprendizaje para converger a valores más bajos. Como se puede observar esto no ocurre, ya que en la experimentación que se muestra no se aplicó el concepto del "learning rate decay"

13.3.9.Resumen de la experimentación con CNN:

Modelo	Nº capas convolucionales	Neuronas por capa totalmente conectada	Función activación	Learning rate	Tamaño mini-batch	Función minimización error	Ciclos
CNN_MNIST	2	500	ReLU	0'001	100	Adam Optimizer	2000
CNN_CIFAR10	2	1000	ReLU	0'1 (Learning Rate Decay)	100	Descenso de gradiente	20000

Modelo	Tasa acierto	Tiempo requerido
MLP_MNIST	0.991	-2 horas
MLP_CIFAR10	0.851	- 6 horas

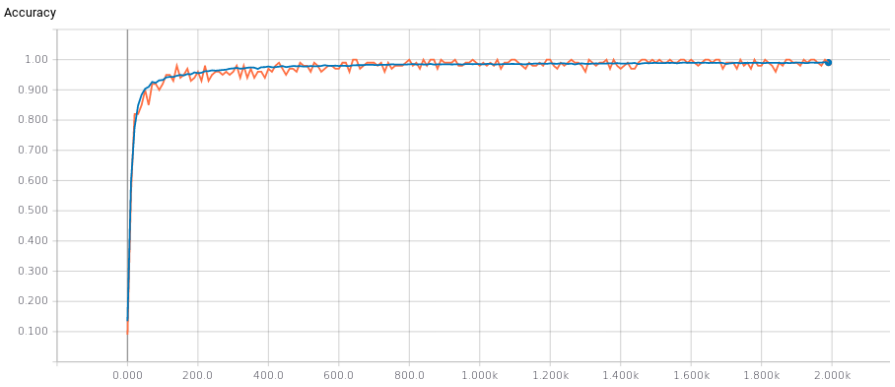

Figura 58. CNN_MNIST

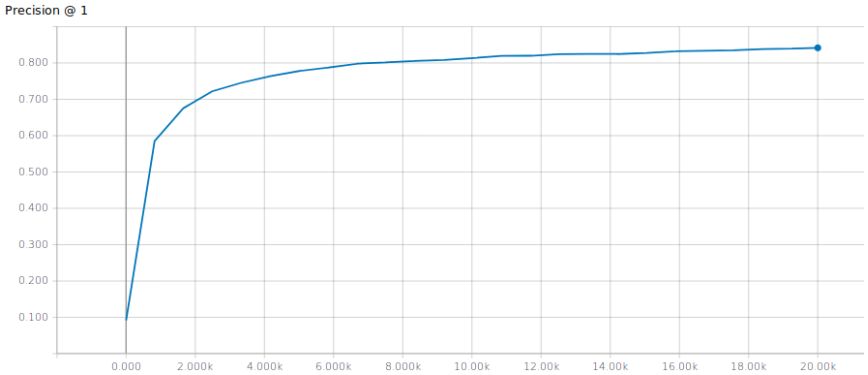

Figura 59. CNN_CIFAR10

14. COMPARATIVA ENTRE COMPORTAMIENTO DE MLP Y CNN

En este apartado se pasará a realizar un análisis comparativo entre las diferencias en cuanto a resultados y rendimiento de los dos tipos de redes con las que se ha trabajado en los anteriores apartados.

Como se ha resaltado en anteriores ocasiones, la motivación de este trabajo era conocer las razones por las que el MLP, aunque sea un aproximador universal, no consigue alcanzar las tasas de acierto a las que llegan las CNN's. Si bien es cierto que esta ultima cuenta con una capa final la cual simula el comportamiento de un MLP (capa completamente conectada), los resultados están francamente distanciados.

A continuación, se muestra una tabla con los dos modelos, y sus mejores resultados de porcentaje de acierto en cada uno de los dominios.

14.1. MLP vs CNN en MNIST:

Como bien se ha analizado, el problema de MNIST está completamente resuelto. Tanto MLP como CNN son capaces de alcanzar altas tasas de de acierto, aun cabe destacar algunas ventajas de un modelo respecto del otro.

Si bien es cierto que las CNN superan el 99% de precisión, sus tiempos de ejecución son sumamente altos en comparación con los del MLP. Sería necesario poner en una balanza el tiempo frente a la tasa de precisión que se desea alcanzar. Cuando se observan los resultados utilizando perceptrón multicapa, estos pueden llegar a alcanzar un 98% en test, pero con un coste computacional y de tiempo relativamente bajo. Hay que tener en cuenta que la cantidad de pesos con los que se trabaja en un MLP queda muy por debajo de los de una red convolucional.

Al otro lado de esta comparación, las CNN superan el 98% de precisión en test anteriormente mencionado en los MLP. Esta diferencia no es tan acusada como se espera de dos modelos tan distintos a nivel de arquitectura, y esto nos lleva a afirmar que el dominio sobre el que se está trabajando tiene una dificultad relativamente baja. En cuanto a los tiempos de ejecución se refiere encontramos, como ya se ha comentado, diferencias abismales.

No tiene demasiado sentido darle a cada modelo el mismo número de iteraciones, debido precisamente a que cada uno necesita tiempos distintos para alcanzar la convergencia. Lo que sí que comparten ambas redes son los tamaños de mini-batch, en ambos casos de 100 imágenes.

Para converger a su máxima precisión, el coste de una CNN al enfrentarse al MNIST esta descompensado. ¿Realmente merece la pena este tipo de redes para un dominio así? Francamente no lo parece, a no ser que se necesiten alcanzan unas tasas de acierto algo más altas, pero no demasiado diferenciadas. Si quisiésemos aplicar este dominio a algún problema del mundo real, como pueda ser el paso de texto manuscrito a texto plano, de cada 10.000 caracteres que reciba la red, clasificará mal unos 100, contra los aproximadamente 180 que confundiría un MLP.

14.2. MLP vs CNN en CIFAR10:

Llegados a este punto, se ha podido comprobar que es razonable evitar el uso de MLP's sobre dominios del tipo CIFAR. Basando la comparación en el análisis realizado a ambos modelos, las diferencias son irrebatibles. En primer lugar la comparativa entre las tasas de acierto arrojadas por cada uno de los modelos son grandiosas. Como se ha mostrado en el apartado del análisis, las CNN superan el 84% de precisión en test, tasa la cual los MLP les cuesta alcanzar incluso en cuanto a entrenamiento se refiere. El código libre que ofrece tensorflow en su página oficial [27] nos muestra unos resultados del 86% de tasa de acierto en test.

Si nos fijamos en el conjunto de test los resultados del MLP dejan mucho que desear. El modelo que mejores valores proporcionó fue una red de 3 capas ocultas, cada una de ellas con 800 neuronas, con la función Leaky_ReLU, intentando así evitar la "muerte neuronal", y minibatches de 100 imágenes. Incluso se realizaron esfuerzos por ajustar las desviaciones estándar con las que los pesos de la red son inicializados aleatoriamente. Todos los esfuerzos por emular el comportamiento de una CNN fueron en vano teniendo en cuenta que alcanzaba, en determinados picos, un acierto del 67%. Sin embargo, se han conseguido mejorar en cierta medida los resultados de un MLP sobre este dataset.

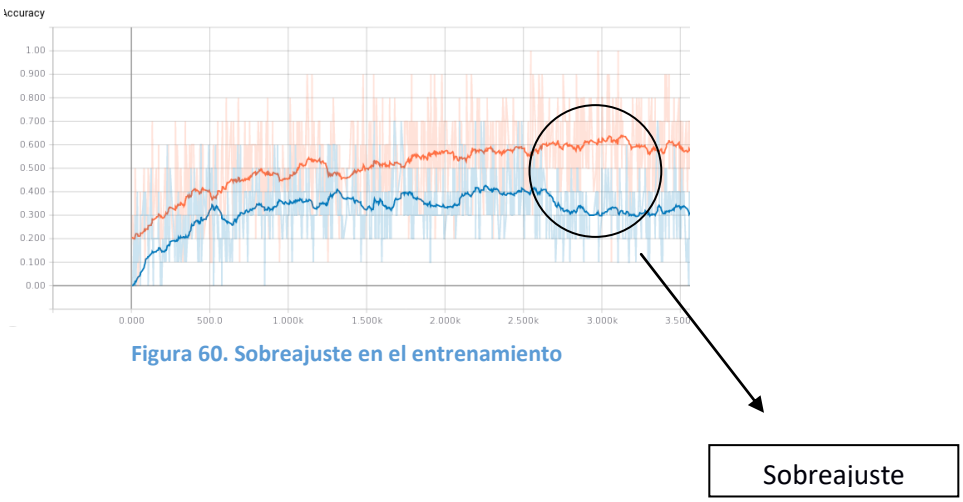

Figura 60. Sobreajuste en el entrenamiento

A parte de las numerosas trabas para alcanzar esta tasa hay que añadir el sobreaprendizaje que se puede observar, aunque no es demasiado acusado. Este tipo de redes, cuando trabaja con dominios con tantas características, no son capaces de extraerlas en su totalidad, y al aumentar el número de ciclos en busca de un mejor resultado, no mejoran, más bien comienza a diferenciarse el acierto de entrenamiento y de test. Otra de las causas que acentúa este comportamiento es la elevada diferenciación que se encuentra en las imágenes de CIFAR, al contrario que en el MNIST, en el que los números manuscritos guardan bastante relación. Las CNN, a diferencia, no parecen presentar ningún tipo de sobreajuste, a no ser que el número de ciclos sea tan elevado que lleve a ese punto.

Entrar en el campo de los tiempos de ejecución no tiene demasiado sentido al comparar MLP y CNN sobre CIFAR10. Si bien es cierto que las redes convolucionales añaden un coste computacional mucho más alto que las MLP, este está justificado sobre el dominio, al igual que con MNIST no. Ambos modelos tienen unos tiempos de ejecución notablemente superiores comparándolo con dominios más simples, sin embargo, para el mismo número de ciclos, las diferencias entre tasas de acierto de uno y otro modelo son enormes. El tiempo que requiere la CNN para alcanzar la tasa más alta conseguida por el MLP es menor, teniendo en cuenta eso mismo, que el porcentaje de acierto no supera el 67%.

15. CONCLUSIONES

En el siguiente apartado se pasan a presentar las conclusiones extraídas a lo largo tanto de la experimentación realizada con los diferentes modelos neuronales, como del estudio previo y la investigación.

15.1. Conclusiones abstraídas de la experimentación general

Todo este trabajo de investigación se ha centrado concretamente en dos modelos y dos dominios: Perceptrón Multicapa y Redes Convolucionales, los cuales trabajaban sobre MNIST y CIFAR10.

En primer lugar, hay que decir que, aunque estos modelos y dominios fuesen el motivo principal del estudio, ya se tenían nociones de como ambos podían llegar a funcionar a lo largo de la experimentación. Por lo tanto no se han encontrado numerosas novedades a lo largo del estudio de los mismos. Sin embargo, si es cierto que se han adquirido valiosos conocimientos en el campo de la inteligencia artificial, más concretamente las redes neuronales, Por otro lado, aunque el comportamiento de estas fuera algo esperado, han surgido nuevas ideas acerca de las mismas.

Cabe destacar la importancia que se ha podido observar en las arquitecturas de las redes neuronales. De primera mano habría que resaltar que una arquitectura inicial condiciona mucho las experimentaciones. Si esta no es escogida adecuadamente, la tarea de experimentación puede ser algo ardua. Por ejemplo, si nos centramos en un MLP, se observa que con tan solo una capa oculta, con un número de neuronas entre un rango de 500 y 1000, es suficiente para alcanzar resultados muy óptimos. Por lo tanto, la parte de investigación que se realizó de acuerdo a la elección de la arquitectura inicial dio sus frutos. Si se retrocede a la parte de la elección de la misma, en el Apartado 12.1., se dijo que una de las aproximaciones que podía tener éxito era la del método Thumb, el cual aconsejaba el uso de un número de neuronas ocultas aproximadamente igual que dos tercios del volumen total de entrada.

Sin embargo, se ha podido observar que esta elección de la arquitectura es muy dependiente del dominio con el que se trabaje. Si pasásemos al CIFAR10, esta misma arquitectura no funciona como se esperaba. Para poder alcanzar unos resultados un poco mejores aumentando el número de ciclos de ejecución se ha visto que los MLP con 3 capas ocultas funcionan ligeramente mejor, sin embargo la afirmación de que el MLP obtiene mejores resultados a largo plazo sobre CIFAR10 con más de una capa oculta no es del todo clara. Basándonos en lo abordado en el Apartado 12.1, donde se explican las ventajas e inconvenientes del uso de una o varias capas ocultas, se puede ver que hay diferentes vertientes en cuanto el uso de varias capas ocultas.

Una vez elegidas las diferentes arquitecturas, las funciones de activación, tasas de aprendizaje y tamaños de mini-batch también influyen, en ocasiones, de manera drástica. Tanto las tasas de aprendizaje como las funciones de activación se han comportado dentro de los rangos normales sobre la documentación que se había leído antes de realizar el trabajo. Sin embargo, aunque de ello también hay información, los tamaños de mini-batch han arrojado resultados algo inesperados. En muchas de las experimentaciones, se decide por tamaños de batch considerablemente más grandes que con los que se ha trabajado en este análisis. Sin embargo, este tipo de batches no añaden, en principio y por lo que se ha podido observar, ningún valor adicional. Simplemente parece que aumenta el tiempo de ejecución del entrenamiento, pero no aporta ninguna mejora a la hora de aciertos sobre el conjunto de test.

Sobre las funciones de activación, el estudio ya reflejaba de una manera bastante cercana a la realidad los resultados. La función de activación ha quedado demostrado que está íntimamente ligada al tipo de dominio con el que se trabaja. Se sabía en un principio que las funciones de tipo sigmoidal no iban a añadir ningún valor a las redes, sino todo lo contrario. La función ReLU ha sido la gran favorita en la mayoría de las experimentaciones por razones obvias. Ya se podía observar en otras experimentaciones su gran eficacia a la hora de tratar problemas de clasificación de imágenes en ambos modelos. Por otra parte, surgió a lo largo de la experimentación una nueva idea. Si se basa la lección de la función de activación en el estudio que se realizó previamente, en el apartado en el cual se hablaba sobre la función ReLU se afirmaba que esta funcionaba de una manera muy satisfactoria, pero que en su contra jugaba el hecho de que, al ser una función que puede llegar a devolver 0, influya en la muerte de neuronas las cuales queden inservibles para el resto del entrenamiento. ¿Porqué no forzar a que esta salida no pueda ser 0? De ahí apareció la función Leaky_ReLU, la cual aplicaba esto mismo. Con la experimentación se pudo comprobar una ligera mejora en cuanto a resultados, sin embargo estas no fueron tan grandes como se esperaba. Si bien es cierto que esta muerte neuronal jugaba un gran papel en el entrenamiento, se pensaba que la utilización de esta función mejoraría de manera notoria los valores devueltos por la red.

En cuanto a las neuronas de salida se refiere, la función de softmax es siempre una apuesta ganadora, generando porcentajes de pertenencia a cada una de las clases, la cual la hace idónea para este tipo de tareas.

El ratio de aprendizaje es quizás uno de los parámetros que más puede influir en los entrenamientos de las redes neuronales, afectando a los tiempos de ejecución y a la precisión de los modelos. Se ha podido comprobar que está muy ligado al tamaño del mini-batch, debido que a medida que este aumenta, la tasa tiene que ir pareja con él, y viceversa. La relación que tiene

con el tipo de arquitectura es también muy estrecha. Cuanto más complejo es el modelo, como puede ser el caso de las CNN, en el que la cantidad de pesos y parámetros es altísima, se necesita un ratio de aprendizaje mucho menor, para evitar grandes cambios en esos pesos y favoreces la convergencia a valores de acierto altos. Sin embargo, en el caso del MLP, aunque se trabaje con un dominio u otro, no es necesaria una modificación tan grande entre ratio de aprendizaje, como cuando se compara un MLP con una CNN. Si bien es cierto que para trabajar con el dominio CIFAR10 es necesario reducir ligeramente este ratio, no es una diferencia tan acusada como cuando se pasa analizar una CNN.

Por último, pero no menos importante, se ha de realizar una importante mención a un parámetro que en ningún momento se tuvo intención de analizar: La desviación estandar en la inicialización aleatoria de pesos dentro del MLP. No cabe mencionar nada especial acerca de este parámetro cuando se habla de MNIST, pero es fue sorprendente comprobar el peso que tiene en CIFAR10. Se pudo comprobar que, mientras el MLP conseguía aprender con una tasa de 0.001 sobre CIFAR10, al modificar la desviación estandar aumentándola, esa misma tasa hacía que el modelo no fuese capaz de avanzar. Es más, era necesario reducirla a menos de 0.00001 para que la red fuera capaz de extraer patrones, y aun así no lo conseguía con la misma eficacia que con rangos menores de esta desviación estandar. Este punto podría ser objeto de un estudio mucho más profundo, analizando los pesos y su evolución desde el inicio con diversos tipos de desviaciones.

Así mismo añadir que, aunque era un resultado esperado, parece que el MLP es el modelo indicado para la clasificación de imágenes en MNIST, mientras que las CNN funcionan infinitamente mejor con CIFAR10. Fue una buena idea pensar, a la hora de diseñar un modelo convolucional, en la extracción de características. Parece una de las mayores razones por las que el MLP no consigue extraer apenas información de imágenes a color. Si se comparan ambos dominios, es obvio que los números manuscritos guardan mucha más relación entre ellos que las imágenes de coches, gatos, aviones etc. Quiere decir que MNIST es un dataset más sencillo, y por lo tanto las MLP también.

15.2. Resultado obtenidos en la experimentación con MLP.

A pesar de no haber obtenido el mismo comportamiento sobre CIFAR10 con MLP que con CNN, se han podido mejorar sus resultados. En las primeras experimentaciones, aunque no reflejadas en este trabajo ya que eran simples pruebas iniciales, no se conseguía superar un 47% de porcentaje de acierto. Teniendo en cuenta esos datos, unos aciertos máximos de más de un 67%, y una convergencia de una media de 62% hacen que los esfuerzos invertidos hayan tenido

sus frutos. De todos modos, estos resultados no son suficientes. Un modelo con una arquitectura de MLP no es conveniente para un dataset de la complejidad del CIFAR10.

¿Entonces sería correcto afirmar que un MLP es un aproximador universal? Puede que, en un principio, cuando se diseñó allá por la década de los 70`s en la que la cantidad de dataset y problemas a abordar no fuera tan grande, si que se pudiese concebir la idea de que un MLP podría encontrar solución a cualquier problema no lineal de manera teórica. Sin embargo, cuando se quieren aplicar este tipo de modelos sobre el mundo real, la afirmación anterior se viene abajo. El MLP fue una buena base para comenzar el desarrollo de las redes de neuronas artificiales, y numerosas arquitecturas, como bien pueden ser las CNN, las cuales incorporan este modelo en la parte final de su arquitectura. Se ha podido constatar que su potencia computacional y de resolución de problemas quedó atrás, dejando paso a un nuevo abanico de nuevas redes.

15.3. Trabajo futuro

A lo largo de todo el estudio realizado en este trabajo de investigación se han podido comprobar que hay ciertos conceptos en los que sería muy interesante hacer hincapié y futuras investigaciones sobre los mismos.

Uno de estos conceptos es la “muerte neuronal”, explicada en anteriores apartados, la cual por medio de la función Leaki_ReLU se ha intentado evitar. Sería interesante a partir de datos extraídos de los modelos, como por ejemplo el Sparsity, realizar un estudio más profundo evitando en mayor medida este problema tan común en la red de neuronas artificiales.

Otro de los parámetros de los que no se esperaba que tuviese un efecto tan directo en el entrenamiento, es la desviación estándar en la aleatorización inicial de los valores de los pesos. Gracias a una correcta adecuación de este valor se puede mejorar el comportamiento inicial de la red neuronal y un estudio basado en esto mismo podría implicar en algún avance en lo que a resultados se refiere.

Por último, uno de los grandes problemas a la hora de entrenar la red de neuronas artificiales es el tiempo que requieren. Se podría contemplar en un trabajo futuro el uso de GPU´s, las cuales otorgan una potencia de procesamiento adicional que sería de gran ayuda en la reducción de los tiempos de ejecución.

16. PLANIFICACIÓN

A continuación se pasa a mostrar la planificación que en un principio se pretendía seguir a lo largo de todo el desarrollo del trabajo de investigación, y por otro lado la planificación real que se acabó siguiente, con el fin de aportar una idea de cómo se han ido repartiendo las tareas a realizar, y los problemas que han surgido en este proceso.

Planificación estimada:

PLANIFICACIÓN PREVISTA

Id	Tarea	Duración (semanas)	Fecha Inicio	Fecha Fin
1	ELECCIÓN DE LOS DOMINIOS	5	06.11.17	08.12.17
2	ELECCIÓN DE LOS MODELOS NEURONALES	4	11.12.17	19.01.18
3	ELECCIÓN DEL LENGUAJE DE PROGRAMACIÓN	3	22.01.18	09.02.18
4	ADQUISICIÓN DE CONOCIMIENTOS EN EL LENGUAJE ESCOGIDO	5	12.02.18	16.03.18
5	PROGRAMACIÓN DE LOS MODELOS NEURONALES	2	19.03.18	6.04.18
6	INVESTIGACIÓN	3	06.04.18	26.04.18
7	EXPERIMENTACIÓN	5	26.04.18	01.06.18
8	ELABORACIÓN DE LA MEMORIA	5	28.05.18	29.06.18
9	REVISION	3	14.09.18	21.09.18

Con el paulatino desarrollo del trabajo, apareció el mayor de los inconvenientes, la carga del dataset CIFAR10 sobre el MLP. En internet no se haya apenas información acerca de esto y por lo tanto no sirvió de mucha ayuda. Ya se ha comentado en el apartado como se tuvo que resolver este problema, lo que causo un desajuste en los tiempo de realización de la investigación. Sumado a eso aparecen otro tipo de factores como que la experimentación y la elaboración de la memoria se demoraron más de lo previsto.

Planificación real:

PLANIFICACIÓN REAL

Id	Tarea	Duración (semanas)	Fecha Inicio	Fecha Fin
1	ELECCIÓN DE LOS DOMINIOS	4	06.11.17	01.12.17
2	ELECCIÓN DE LOS MODELOS NEURONALES	4	04.12.17	12.01.18
3	ELECCIÓN DEL LENGUAJE DE PROGRAMACIÓN	3	15.01.18	02.02.18
4	ADQUISICIÓN DE CONOCIMIENTOS EN EL LENGUAJE ESCOGIDO	4	05.02.18	02.03.18
5	PROGRAMACIÓN DE LOS MODELOS NEURONALES	2	05.03.18	23.03.18
6	RESOLUCIÓN PROBLEMA CARGA DATOS	4	19.03.18	20.04.18
7	INVESTIGACIÓN	5	09.04.18	11.05.18
8	EXPERIMENTACIÓN	6	23.04.18	01.06.18
9	ELABORACIÓN DE LA MEMORIA	6	28.05.18	06.07.18
10	REVISION	2	10.09.18	21.09.18

17. PRESUPUESTO TOTAL DEL PROYECTO

A continuación se va a exponer los costes que se han producido en la realización del proyecto, tanto a nivel software, hardware, como también a nivel personal. En este último, a parte, se ha decidido incluir la tarea del supervisor del trabajo, el cual también ha sido necesario destacar.

Recursos hardware/software:

A la hora de calcular los recursos a nivel de software y de hardware se han decidido incluir el propio material hardware con el que se ha realizado la experimentación, como es el ordenador portátil, y recursos los cuales ha surgido su adquisición a lo largo del desarrollo del trabajo, debido a problemas a lo largo del mismo, como pueda ser el cable USB o el receptor wifi USB.

Recurso	Coste
ORDENADOR PORTATIL HP HEWLETT - PACKARD i5	800 €
CABLE USB	3 €
MICROSOFT OFFICE 2016	80 €
RECEPTOR WIFI USB	15 €
COSTE TOTAL	**898 €**

Recursos humanos:

En base a la planificación anteriormente mostrada, se han extraído el número de horas tanto de la parte del investigador como de la del supervisor:

Personal	Horas	Costes	
		€/hora	Total €
INVESTIGADOR	403,50	13 €	5.246 €
SUPERVISOR	60,00	19 €	1.140 €
		COSTE TOTAL	**6.386 €**

Presupuesto total:

A continuación se muestra un resumen con el coste total del proyecto:

RESUMEN COSTES	
RECURSOS HUMANOS	6.386 €
HARDWARE/SOFTWARE	898 €
COSTE TOTAL	**7.284 €**

18. REFERENCIAS

[1] Definición de Machine learning según Arthur Samuel, https://www.ibm.com/developerworks/community/blogs/jfp/entry/What_Is_Machine_Learning?lang=en

[2] Big Data, https://searchdatacenter.techtarget.com/es/definicion/Big-data

[3] Crecimiento exponencial de los datos, https://www.eetimes.com/author.asp?section_id=36&doc_id=1330462

[4] Ley de Moore - https://www.britannica.com/technology/Moores-law

[5] Gráfica Ley de Moore - https://www.geeknetic.es/Editorial/1406/La-realidad-sobre-los-nanometros-en-procesos-de-fabricacion-de-CPUs-y-GPUs.html

[6] CVL - https://iron-ai.com/blog/2017/08/15/machine-learning-no-big-data-vender/

[7] Aplicación de redes de neuronas artificiales en la detección de enfermedades oculares, https://www.forbes.com/sites/samshead/2018/08/13/google-deepminds-ai-can-detect-50-eye-disease-conditions-and-save-sight/

[8] Neurona biológica, http://en.citizendium.org/wiki/Neuron

[9] Red neuronal del cerebro humano, https://www.oreilly.com/library/view/neural-networks-and/9781492037354/ch01.html

[10] Comparativa entre neuronas biológica y neuronas artificiales, https://medium.com/@ivanliljeqvist/the-essence-of-artificial-neural-networks-5de300c995d6

[11] Aplicaciones generales de las redes de neuronas artificiales, http://publicaciones.unisimonbolivar.edu.co/rdigital/ojs/index.php/identic/article/view/1489/1425

[12] Aprendizaje hebbiano, *http://redesneuronal.blogspot.com/*

[13] Perceptrón Simple, https://nasirml.wordpress.com/2017/11/19/single-layer-perceptron-in-tensorflow/

[14] Perceptrón Multicapa, https://www.mdpi.com/2078-2489/3/4/756

[15] Función ReLU, https://www.cs.toronto.edu/~guerzhoy/411/lec/W04/activationfn.pdf

[16] Función Softmax, https://elvex.ugr.es/decsai/deep-learning/slides/NN8%20Softmax.pdf

[17] Salida de la funciónSoftmax, https://towardsdatascience.com/deep-learning-concepts-part-1-ea0b14b234c8

[18] Arquitectura de una CNN, https://www.ibm.com/developerworks/library/cc-machine-learning-deep-learning-architectures/index.html

[19] Parameter Sharing, http://cs231n.github.io/convolutional-networks/

[20] MNIST, http://yann.lecun.com/exdb/mnist/

[21] CIFAR10, https://www.cs.toronto.edu/~kriz/cifar.html

[22] Teorema de la Aproximación Universal, http://bdigital.unal.edu.co/34785/1/34992-136592-1-PB.pdf

[23] Tamaños de batch vs tasas de acierto y numero de ciclos, https://pdfs.semanticscholar.org/43c9/6ccaa90b3875ce2912063b9949716f8d5824.pdf

[24] Adam Optimizer, https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/.

[25] RMSProp, http://ruder.io/optimizing-gradient-descent/index.html#rmsprop

[26] Cross Entropy, https://www.tensorflow.org/api_docs/python/tf/losses/softmax_cross_entropy

[27] CNN Tensorflow, https://www.tensorflow.org/tutorials/images/deep_cnn